Performance Analysis and Design of a Discreet Cosine Transform processor Using CORDIC algorithm

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

in

VLSI Design and Embedded System

By

Satyasen Panda

ROLL No: 208EC2o8

Under the Guidance of

Prof. K.K.Mahapatra

Department of Electronics and Communication Engineering

National Institute Of Technology Rourkela

2008-2010

National Institute Of Technology Rourkela

C E R T I F I C A T E

This is to certify that the thesis entitled, **"Performance Analysis and Design of a Discreet Cosine Transform processor using CORDIC algorithm"** submitted by **Satyasen Panda** in partial fulfillment of the requirements for the award of Master of Technology Degree in **Electronics & Communication Engineering** with specialization in **"VLSI Design and Embedded System"** at the National Institute of Technology, Rourkela is an authentic work carried out by him under my supervision. To the best of my knowledge, the matter embodied in the thesis has not been submitted by him to any other University / Institute for the award of any Degree or Diploma.

Date:

Prof. K.K.Mahapatra
Dept. of Electronics & Communication Engineering
National Institute of Technology
Rourkela-769008

Acknowledgement

First of all, I would like to express my deep sense of respect and gratitude towards my advisor and guide **Prof. K. K. Mahapatra,** who has been the guiding force behind this work. I want to thank him for introducing me to the field of Digital Design and giving me the opportunity to work under him. I consider it my good fortune to have got an opportunity to work with such a wonderful person.

I express my respects to Prof. S.K. Patra, Prof. G. S. Rath, Prof. S. Meher, Prof. S.K.Behera, Prof. Punam Singh, Prof. D.P.Acharya, Prof. S.K. Das, Prof. Murthy and Prof. Ari for teaching me and also helping me how to learn. They have been great sources of inspiration to me and I thank them from the bottom of my heart.

I would like to thank all faculty members and staff of the Department of Electronics and Communication Engineering, N.I.T. Rourkela for their generous help in various ways for the completion of this thesis.

I would like to thank my friends. I am also thankful to my classmates for all the thoughtful and mind stimulating discussions we had, which prompted us to think beyond the obvious.

Satyasen Panda

Contents

page no

ABSTRACT

CORDIC is an acronym for COrdinate Rotation Digital Computer. It is a class of shift adds algorithms for rotating vectors in a plane, which is usually used for the calculation of trigonometric functions, multiplication, division and conversion between binary and mixed radix number systems of DSP applications, such as Discreet cosine Transform(DCT). The Jack E. Volder's CORDIC algorithm is derived from the general equations for vector rotation. The CORDIC algorithm has become a widely used approach to elementary function evaluation when the silicon area is a primary constraint. The implementation of CORDIC algorithm requires less complex hardware than the conventional method.

In digital communication, the straightforward evaluation of the cited functions is important, numerous matrix based adaptive signal processing algorithms require the solution of systems of linear equations, the computation of eigen values, eigenvectors or singular values. All these tasks can be efficiently implemented using processing elements performing vector rotations. The (CORDIC) offers the opportunity to calculate all the desired functions in a rather simple and elegant way. Due to the simplicity of the involved operations the CORDIC algorithm is very well suited for VLSI implementation. The rotated vector is also scaled making a scale factor correction necessary. VHDL coding and simulation of selected CORDIC algorithm for sine and cosine, the comparison of resultant implementations and the specifics of the FPGA implementation has been discussed.

In this thesis, the CORDIC algorithm has been implemented in XILINX Spartan 3E FPGA kit using VHDL and is found to be accurate. It also contains the implementation of Discrete Cosine Transform using radix-2 decimation-in-time algorithm in Xilinx. on the same FPGA kit. Due to the high speed, low cost and greater flexibility offered by FPGAs over DSP processors the FPGA based computing is becoming the heart of all digital signal processing systems of modern era. Moreover the generation of test bench by Xilinx ISE 9.2i verifies the results with directly computed dct values from mat lab.

List of figures

page no

List of tables

page no

Chapter 1

Introduction

1.1MOTIVATION

The advances in IC technology have great interests in developing special purpose, parallel processor arrays such systolic arrays have been extensively used. The basic arithmetic computation of these parallel arrays has often been implemented with a MAC ,because these operations arise in DSP applications. The reduction in hardware cost also motivated the development of sophisticated signal processing algorithms which need the evaluation of functions such as trigonometric and logarithmic functions, which cannot be evaluated efficiently with MAC based arithmetic units. So when signal processing algorithms incorporate these elementary functions, it is sure to observe significant performance failure.

So an arithmetic computing algorithm known as CORDIC (Coordinate Rotation Digital Computer) has received great attention, as it offers an iterative formulation to efficiently calculate each of these elementary functions. Specially, all the evaluation tasks in CORDIC are formulated as a rotation of vectors in various Coordinate systems. By varying a few parameters, the same CORDIC processor is capable of iteratively calculates these elementary functions using the same hardware in the same amount of implementation of pipelined VLSI array processors.

2-D DCT algorithms are the mostly used for image compression, the main focus of this thesis will be on the efficient hardware implementations of 2-D DCT based image

compression by reducing the number of computations, increasing the accuracy of reconstruction, and reducing the chip area using CORDIC algorithm. This reduces the power consumption of the compression technique. The number of applications that require higher-dimensional DCT algorithms are growing, emphasis will be paid to the algorithms like CORDIC that are extensible to higher dimensional cases.

1.2 Organization of this Thesis

Chapter 2 of this thesis describes fundamentals of CORDIC algorithm and its mathematical formulation. Different approaches have been described and compared with each other. In chapter 3 different implementation methods of CORDIC have been described and their pros & cons have been discussed. In chapter 4 an overview of Discreet Cosine Transform (DCT) has been provided and various properties of DCT have been discussed. In chapter 5 different implementation methods of DCT have been discussed, also how DCT can be implemented using CORDIC is explained. In chapter 6 we have implemented a DCT processor using a CORDIC core. Here first the CORDIC processor is designed and tested, then DCT processor is designed with this CORDIC core and tested with real time data.

Chapter 2

CORDIC : Algorithm for elementary function computation

2.1 CORDIC fundamentals

The CORDIC is hardware-efficient algorithms for computation of trigonometric and other elementary functions that use only shift and add to perform. The CORDIC set of algorithms for the computation of trigonometric functions was designed by Jack E. Volder in 1959 to help building a real-time navigational system for the B-58 bomber. Later, J. Walther in 1971 extended the CORDIC scheme to other functions. The CORDIC method of functional computation is used by most calculators (such as the ones by Texas Instruments and HP) to approximate the normal transcendental functions.

Depending on the configuration, the resulting module implements pipelined parallel-pipelined, word-serial, or bit-serial architecture in one of two modes: rotation or vectoring. In rotation mode, the CORDIC rotates a vector by a certain angle. This mode is used to convert polar to Cartesian coordinates. For eg consider the multiplication of two complex numbers x+jy and $(\cos(\theta) + j\sin(\theta))$ .The result u+jv, can be obtained by calculating the final coordinate after rotating a 2x2 vector $[x \;\; y]^T$ through an angle (θ) and then scaled by a factor r.This is achieved in CORDIC via a three-stair procedure: angle conversion, Vector rotation and scaling.

The basic block diagram of CORDIC processor is given below..

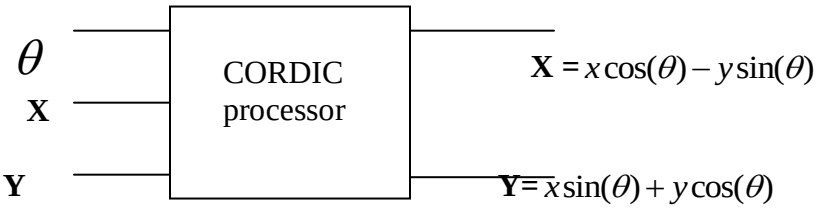


Fig 2.1:CORDIC block diagram

During the angle conversion phase, the angle θ is represented as the sum of a decreasing sequence of elementary angles $\{\delta_{m,i} = 2^{-S_{m,i}}$ *defines a radix* 2 *number system.*, $0 \le i \le b-1\}$ where b is the number of rotations ,which in turn depends on the precision we want, so that

$$\theta = \sum_{i=0}^{b-1} \mu_i \alpha_{m,i}$$

In the above algorithm, the parameters $\mu_i (= \pm 1)$ constitutes an explicit representation of θ, and b is the number of bits in the register. Variable $m \in \{1, 0, -1\}$ represents the rotation in three types of systems: the circular, linear and hyperbolic respectively .In DCT, θ is known from the beginning and the operation is always in Circular system with m=1. The scaling factor $K = \prod_{i=0}^{b-1} \cos(\mu_i \tan^{-1} 2^{-i})$ will be a constant and not dependent of $|\mu_i|$=1.Hence K can be computed in the beginning and by calculating the limit comes to 0.6072512.All the angles in angle conversion contains pre evaluation of arc tan. So this is implemented in the form of a look-up table in hardware system. The circular cordic rotation is shown in figure below.

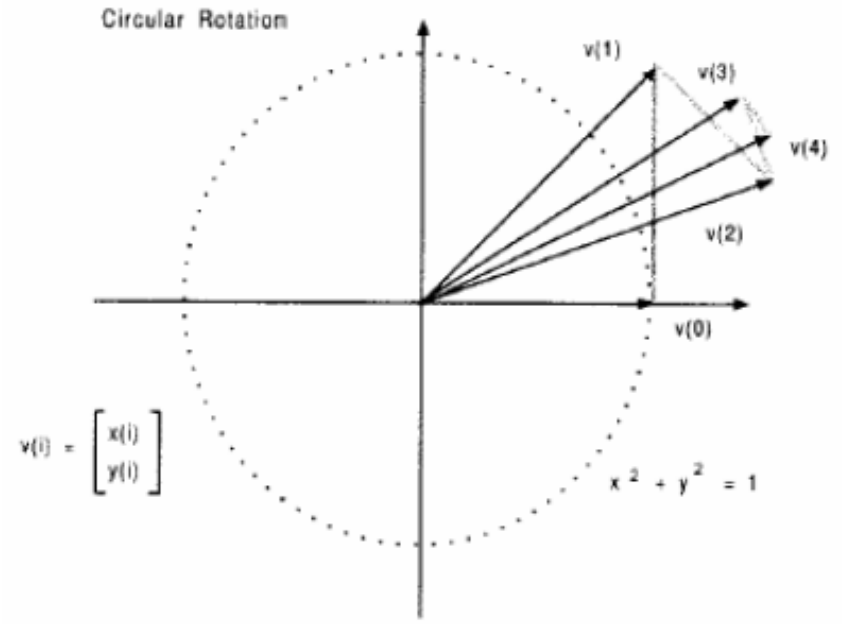


Fig2.2: circular Cordic rotation

The fundamentals of CORDIC algorithm are given below.

2.1.1The Rotation Transform

All the trigonometric functions can be evaluated from functions using vector rotations. The CORDIC algorithm provides an iterative method of doing vector rotations by certain angles using only shifts and add operations. The algorithm is derived using the general rotation transform:

$$\begin{aligned} X' &= X\cos(\phi) - Y\sin(\phi) \\ Y' &= Y\cos(\phi) + X\sin(\phi) \end{aligned} \qquad (1)$$

where (X', Y') are the coordinates of the resulting vector after rotation of a vector with coordinates (X,Y) through an angle of ϕ in the rectangular plane. These equations can be:

$$\begin{aligned} X' &= \cos(\phi).\left[X - Y\tan(\phi)\right] \\ Y' &= \cos(\phi).\left[Y + X\tan(\phi)\right] \end{aligned} \qquad (2)$$

Now, if the rotation angles are restricted such that $\tan(\phi)=\pm 2^{-i}$ then the tangent multiplication term is reduced to a shift operation. Hence angles of rotation can be found by doing a continuously smaller elementary rotations. The above equation for rotation can be expressed as:

$$X_{i+1} = K_i(X_i - Y_i\sigma_i 2^{-i})$$
$$Y_{i+1} = K_i(Y_i + X_i\sigma_i 2^{-i}) \qquad (3)$$

Where Ki= $cos(tan^{-1}(2^{-k}))$ and $\partial_k = \pm 1$ depending upon the earlier iteration. Taking away the scale constant from the equations yields a shift and add algorithm for vector rotation. The product K_i approaches the value of 0.607312. The CORDIC algorithm in its binary version can be expressed as a sequence of three equations as shown:

$$X_{k+1} = \left[X_k - mY_k.\partial_k.2^{-k}\right]$$
$$Y_{k+1} = \left[Y_k + X_k.\partial_k.2^{-k}\right] \qquad (4)$$
$$Z_{k+1} = \left[Z_{k-}.\partial_k.\varepsilon_k\right]$$

Where $m = \pm 1$.

2.1.2Evaluation of Sine and Cosine Functions

To evaluate the *sin* θ and *cos* θ for $\theta \le \pi/2$, we let m = 1, $\varepsilon_k = tan^{-1}(2^{-k})$ and assume:

$$C = \prod_{k=0}^{n} \cos(\varepsilon_k) \qquad (5)$$

Then the equations of the CORDIC algorithm for evaluating sine and cosine functions can be shown as:

$$X_{k+1} = \left[X_k - Y_k.\partial_k.2^{-k}\right]$$
$$Y_{k+1} = \left[Y_k + X_k.\partial_k.2^{-k}\right] \qquad (6)$$
$$Z_{k+1} = \left[Z_{k-}.\partial_k.\varepsilon_k\right]$$

The radix 2 system is taken here because it avoids the use of multiplications while implementing the above equation. Hence a CORDIC iteration can be realized using shifters and adders only.The lower figure shows the structure of a processing element which implements one CORDIC iteration.

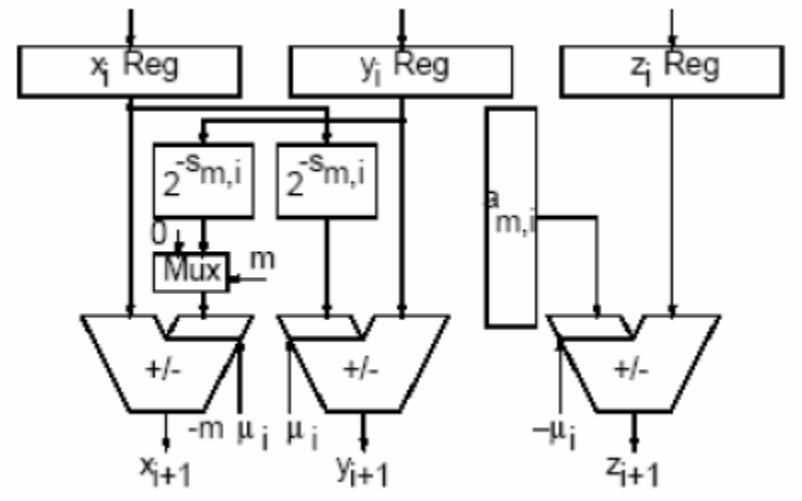


Fig 2.3: structure of a processing element for one CORDIC iteration

The rotation mode and vectoring mode are two schemes for the CORDIC algorithm. In rotation mode, the aim is to rotate the given input vector $(x , y)^t$ with a given angle. After n nos of iterations, z_n is driven to zero and the total accumulated rotation angle is equal to desired angle

However, the CORDIC iteration is not a perfect rotation. It is pointed out that for a fixed-point implementation with data word length of W bits, no more than W CORDIC iterations required to be performed. The large number of iterations decreases its speed performance in a measure way.Also the algorithm computational accuracy needs to be taken into account . For example, the finite precision of the involved variables and the rounding errors; a desired rotation angle θ can only be considered by the n rotational angles so that the accuracy achieved by the CORDIC rotation is determined .

The angles in design of Discreet Cosine Transform are known . If conventional CORDIC is used, for any angle, no of iterations are limited. Instead of this, if we recode the angle in a proper way such that the sign sequence $\alpha_i = \sin^{-1} 2^{-i} \approx 2^{-i}$, then several micro rotations will be removed. That in turn reduces the nos of iterations and speeds up the CORDIC algorithm. Also to get the correct rotated coordinates, the result must be multiplied with a scaling factor K.If we remove this scaling factor, the complexity of

CORDIC algorithm will be reduced. And also, for a given rotation angle by changing the micro rotation angles from $\alpha_i = \tan^{-1} 2^{-i}$ to $\alpha_i = \sin^{-1} 2^{-i} \approx 2^{-i}$, then we should not create a look up table and the burden of creating hard ROM will be reduced. The large number of iterations decreases its speed performance seriously and also takes large power. Secondly, a scale factor operation is required in order to approve the final coordinate $[X_{f,} Y_f]^T$ has the same norm as the initial coordinate $[X_o, Y_o]^T$.

To remove the disadvantages of the Conventional CORDIC algorithm, a new CORDIC algorithm came out, where it overcomes all the disadvantages of the conventional CORDIC algorithm. This new CORDIC algorithm removes the operation of scaling factor correction as well as decreases the number of CORDIC iterations. The algorithm is derived from the general rotation transform which is in Equation (1). Since sine and cosine functions are present, they can be represented as following using *Taylor Series :*

$$\begin{aligned} \sin(\alpha) &= \alpha - (3!)^{-1}.\alpha^3 + (5!)^{-1}.\alpha^5 + ... \\ \cos(\alpha) &= 1 - (2!)^{-1}.\alpha^2 + (4!)^{-1}.\alpha^4 + ... \end{aligned} \tag{7}$$

The series up to order three is applied to Equation (1) with the correction of coefficients. The new CORDIC algorithm can be summarized below. Equation (8) describes a rotation with a scaling of an intermediate plane vector $v_i = [x_i, y_i]^T$ *to* $v_{i+1} = [x_{i+1}, y_{i+1}]^T$.

For $\alpha_i = \sin^{-1} 2^{-i} \approx 2^{-i}$ with $i \in \{0,.....,n-1\}$, we have

$$\begin{aligned} x_{i+1} &= x_i.\cos(\alpha_i) - y_i.\sin(\alpha_i) \\ &= x_i.(1 - 2^{-1}.\alpha^2) - y_i.(\alpha - 2^{-4}.\alpha^3) \\ &= x_i.(1 - 2^{-2i-1}) - y_i.(2^{-i} - 2^{-3i-4}) \\ y_{i+1} &= y_i.\cos(\alpha_i) + x_i.\sin(\alpha_i) \\ &= y_i(1 - 2^{-1}.\alpha^2) + x_i.(\alpha - 2^{-4}\alpha^3) \\ &= y_i.(1 - 2^{-2i-1}) + x_i.(2^{-i} - 2^{-3i-4}) \end{aligned} \tag{8}$$

$$z_{i+1} = z_i - \mu_i.\alpha_i$$

$$= z_i - \mu_i.2^{-i} \qquad with\ \mu_i \begin{cases} 0\ z_i \le \alpha_i \\ 1\ z_i > \alpha_i \end{cases} and\ i \in \{0,....,n-1\}$$

where

$$\alpha_i = 2^{-i}$$

$$z_{i+1} = z_i - \mu_i.2^{-i}\ with\ i \in \{2,3,....,n-1\} \tag{9}$$

In case of large target rotation angle θ ,if α_i is set too small, the rotation precision will be increased as well as the number of CORDIC iterations required . Hence the precision of the design needs to be traded of with the hardware limitations and other performances.

The new CORDIC algorithm has the rotation angles θ from 0 to 45 degree(s). The value of the start point vector $[x,y]^T$ is $[1,0]^T$. Simulations mat lab show that in maximum rotation number for each rotation angle is fixed to ten.

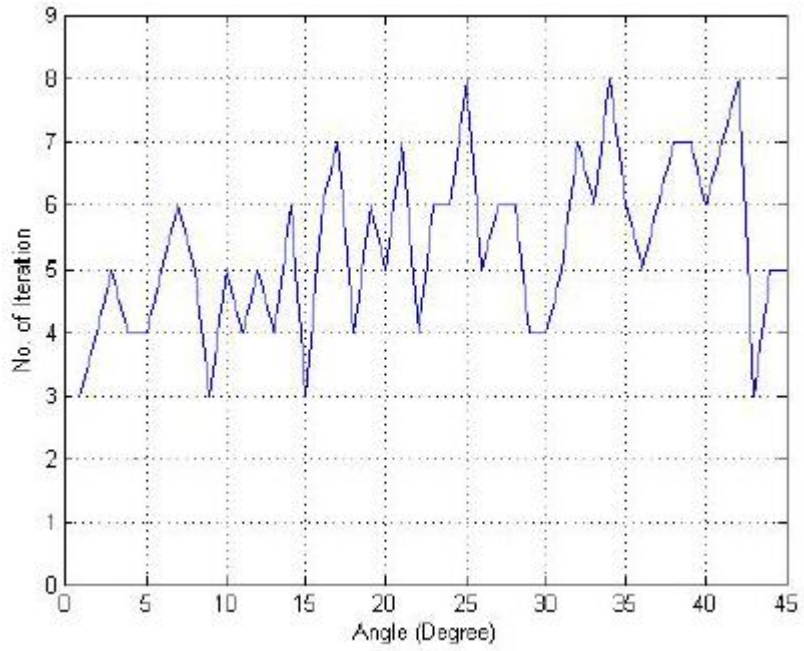


Fig 2.4: No of iterations for angle 0^o to 45^o

However for the new CORDIC algorithm, the input angle is limited to 0^o to 45^o.So if we want to have rotation for other angles less than 90^o, method of "domain folding" must be applied. For example if we want to rotate the angle (θ) between 90^o and 45^o, we have to perform $\phi = \frac{\pi}{2} - \theta$ .Then negative rotation of the θ must be done so that

$$\begin{bmatrix} x'_+ \\ y'_+ \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

So for the equations (8) we have to change '-'to '+' and vice versa. After computing the rotations, the final results are $x_{fd} = y'_+$, $y_{fd} = -x'_+$ .Although we have approximated the angle as right shifting, the error between the result computed by conventional CORDIC and new CORDIC algorithm is very less as shown in the error plot given below.

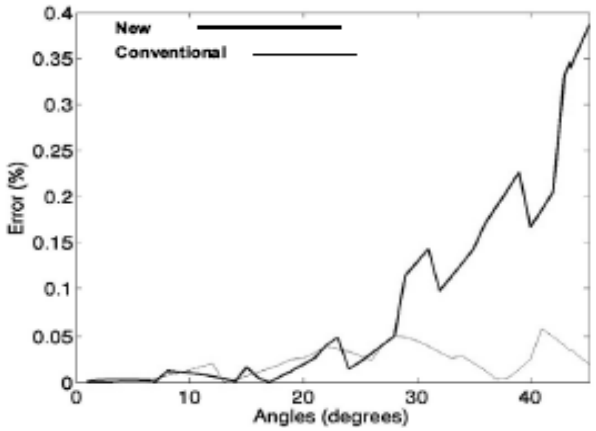


Fig 2.5: Error plot between New and Conventional CORDIC.

Even though the new algorithm is good in terms of lack of scaling factor and reducing the number of iterations, it has got some disadvantages like in the word length as the word length must be taken as 32 bits. It is the optimum word length. Now the design of each iteration of the new Cordic algorithm is given below.

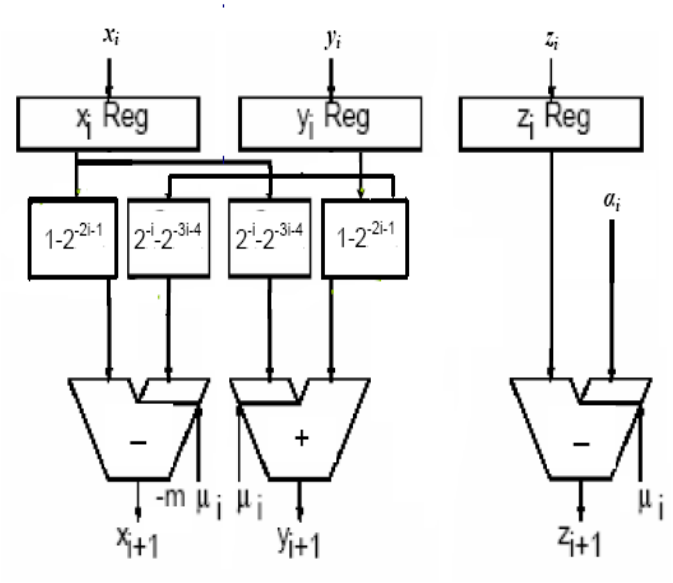


Fig 2.6: Design of a iteration in the new Cordic algorithm

The blocks in between the adders and registers are shifters where the main shifting operation is 32 bit shift operation.

Chapter 3

Different architectures of CORDIC

3.1 Different architectures of CORDIC

There are mainly four types of architectures with which the CORDIC ip core can be designed. They are 1) Iterative word-serial architecture

2) Parallel-Pipelined architecture.

3) Iterative Bit-Serial architecture.

4) Iterative Bit-parallel architecture

Now the description of every architecture is described below..

3.1.1 Word-serial architecture:

Direct implementation of the CORDIC iterative equations provides the block diagram shown in Fig 2.1. The vector coordinates to be converted, or initial values, are loaded via multiplexers into different registers RegX, RegY, and RegA. RegA, along with an adjacent adder/subtractor, multiplexer, and an arctan LUT(look up table), is often called an angle accumulator. Then on every passing clock cycles, the registered values are passed through adders/subtractors and shifters. Each iteration takes one clock cycle, so that in *n* clock cycles, *n* iterations are performed and the converted coordinates are stored in the various registers.

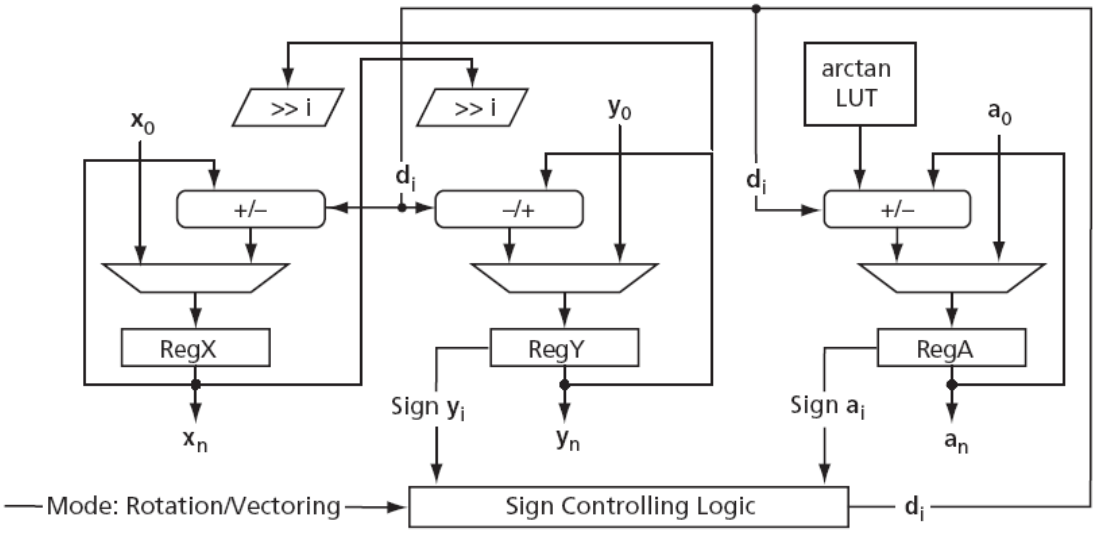


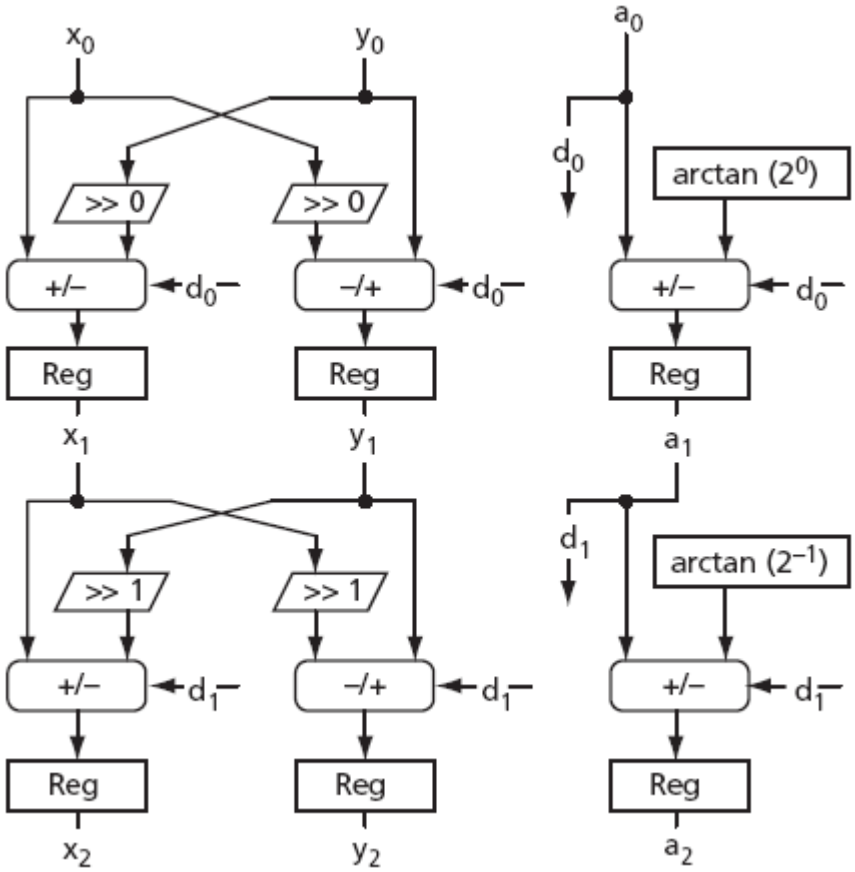
Fig 3.1:Iterative word-serial CORDIC block diagram.

Depending on the CORDIC mode (rotation or vectoring), the sign-controlling logic block performs either the RegY or the RegA sign bit(positive or negative). So that it can decide what type of operation (addition or subtraction) needs to be performed after each iteration. The Look Up Table keeps a pre-computed table of the values. The number of entries in the Look Up Table equals the required number of iterations, *n*.

The iterative word-serial CORDIC algorithm takes *n + 1* clock cycles to complete a single vector coordinate conversion.

3.1.2 Parallel-pipelined architecture:

This architecture represents a version of the sequential CORDIC algorithm . Instead of reusing the same hardware for all iteration stages, the parallel architecture provides a separate processor for every iteration. An example of the parallel CORDIC architecture for rotation mode is shown in fig 2.2



……..

……..

Fig 3.2: Parallel pipelined architecture for CORDIC

Each of the *n* processors present in the block performs a specific iteration, and a particular processor always performs the same iteration. All the shifters perform the fixed shift, so that it can be implemented in the FPGA . Every processor utilizes a individual arctan value that can also be hardwired to the input of every angle accumulator in the absence of a state machine which provides simplicity to this type of architecture.

The parallel architecture is much faster than the sequential architecture described in the "iterative Word-serial architecture "in fig 2.2. It takes new input data and puts out the results at every clock cycle, introducing a latency of *n* clock cycles. The architecture which is used in the design of the 2D DCT is this parallel-pipelined architecture because this architecture which provides high throughput and low power consumption.

3.1.3 Iterative bit-serial architecture:

For the smaller FPGA implementation Where the CORDIC conversion speed is not an issue, this architecture is preferred for performance enhancement. For instance, in order to initialize a Sine/Cosine Look up table, the iterative bit-serial CORDIC is the solution. Fig 2.3 shows the simplified block diagram of the bit-serial architecture. The shift registers get the initial data presented in bit-parallel form, i.e., all bits at once. The data then shifts to the right, before reaching the serial adders/subtractors. Every iteration takes *m* clock cycles, where *m* is the bit resolution. Serial shifters are implemented by properly tapping the bits of the every shift register. The control circuitry provides sign-padding (positive or negative) of the shifted serial data to realize its correct sign extension. The results from the serial adders return back to the shift registers, so that after *m* clock cycles the results of iteration are stored in the shift registers.

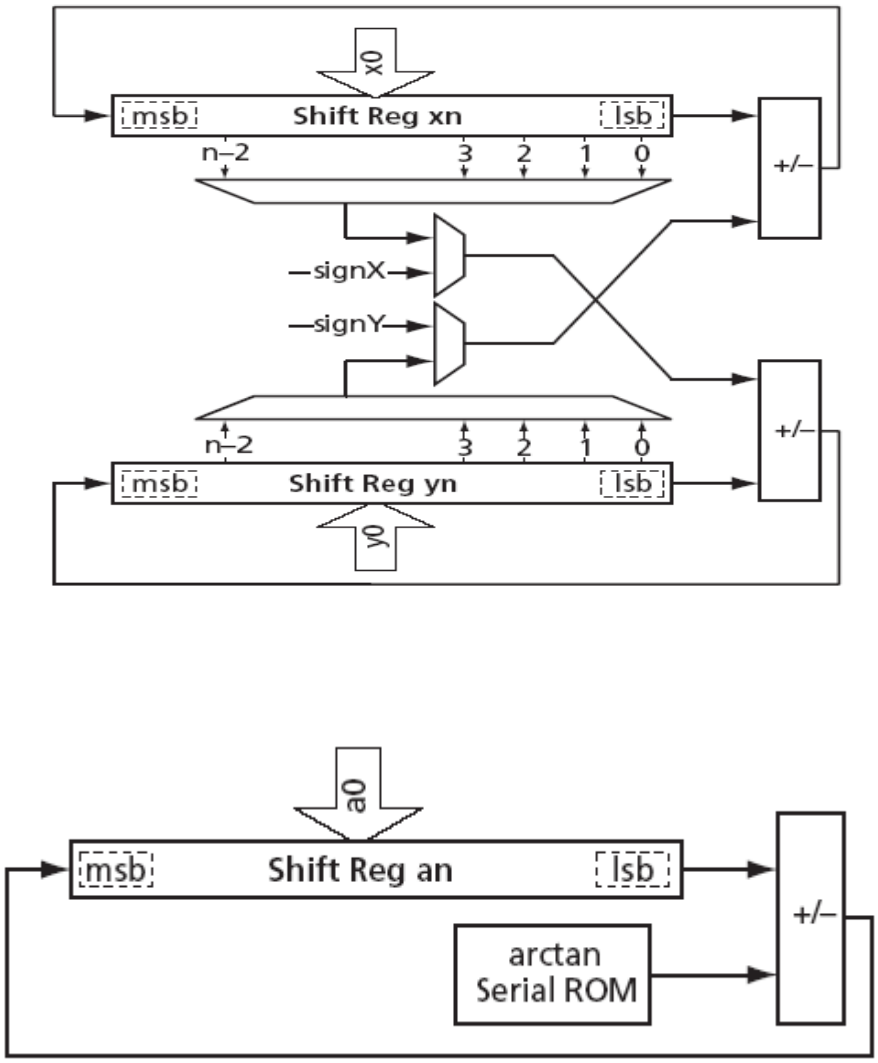


Fig 3.3: Iterative bit-Serial CORDIC architecture.

3.1.4 Iterative bit parallel architecture:

In iterative bit parallel architecture each branch consists of an adder-subtractor combination, a shift unit and a register for buffering the output. At first calculated initial values are fed into the registers by the multiplexer where the MSB of the stored value in the z-branch determines the actual operating mode for the adder-subtractor. Signals in the x and y branch pass the shift units and are then added to or subtracted as required from the unshifted signal in the opposite path.

The z branch arithmetically combines the registers values with the values taken from a lookup table whose address is changed according to the number of iteration. For n numbers of iterations the output is mapped back to the registers before initial values are fed again and the final sine value can be accessed at the output. A simple finite-state machine(FSM) is needed to control the multiplexers, the shift distance and the addressing of the permanent values.

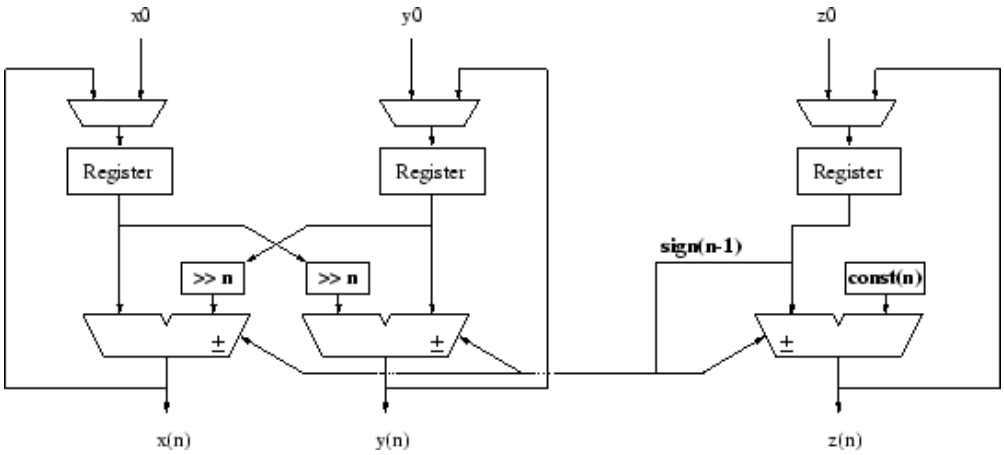


Fig 3.4: Iterative bit parallel architecture.

When implemented in an FPGA the initial values for the vector coordinates as well as the constant values in the LUT(lookup table) can be hardwired . The adders and the subtractors components are carried out separately and a multiplexer controlled according to the sign of the angle accumulator differentiates between addition and subtraction by routing the signals as required. The shift operations as implemented change the shift distance with the number of iterations accordingly. In addition the output

rate is also decreased by the fact that operations are performed iteratively and therefore the maximum output rate equals $\frac{1}{n}$ times the clock rate.

Of all the architectures of the CORDIC are mentioned and described, the architecture used in this project is the parallel pipelined architecture.

Chapter 4

Discrete Cosine Transform
-An overview

4.1 Overview of DCT

Discrete cosine transform (DCT) is widely used in image processing, especially for compression. The Discrete Cosine Transform (DCT) was first proposed by Ahmed et al. (1974), and it has got more importance in recent years Some of the applications of two-dimensional DCT involve image compression and compression of video frames, while multidimensional DCT is mainly used for compression of video streams.

2-D DCT algorithms are the mostly used for image compression, the main focus of this chapter will be on the efficient hardware implementations of 2-D DCT based image compression by reducing the number of computations, increasing the accuracy of reconstruction, and reducing the chip area. This reduces the power consumption of the compression technique. The number of applications that require higher-dimensional DCT algorithms are growing, emphasis will be paid to the algorithms that are extensible to higher dimensional cases.. Since JPEG has some very useful strategies for DCT quantization and compression, it was only developed for low compressions. The 8×8 DCT block size was chosen for speed not for performance.

Like other transforms like DFT, FFT,DST the Discrete Cosine Transform (DCT) trys to decorrelate the image data. After decorrelation each transform coefficient can be encoded independently without losing compression efficiency.So this part describes the DCT and some of its important properties.

4.1.1The One-Dimensional DCT

The DCT definition of a 1-D sequence of length N is

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) \cos\left[\frac{\pi(2x+1)u}{2N} \right] \qquad (1)$$

for $u = 0,1,2,\ldots, N — 1$. The inverse transformation is defined as

$$f(x) = \sum_{u=0}^{N-1} \alpha(u) c(u) \cos\left[\frac{\pi(2x+1)u}{2N} \right] \qquad (2)$$

for x = 0,1,2..,, N — 1. In both equations (1) and (2) $\alpha(u)$ is defined as

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & for \quad u = 0 \\ \sqrt{\frac{2}{N}} & for \quad u \neq 0. \end{cases} \qquad (3)$$

It is clear from equation(1) that for u = 0, $c(u=0) = \sqrt{\frac{1}{N}} \sum_{x=0}^{N-1} f(x)$. Thus, the first transform coefficient is the average value of the all the sample sequences.This value is referred to as the *DC Coefficient.* All other transform coefficients are called the *AC Coefficients.*

The plot of $\sum_{x=0}^{N-1} \cos\left[\frac{\pi(2x+1)u}{2N} \right]$ for N = 8 and varying values of u is shown in Figure 1. The first the top-left waveform (u = 0) offers a constant (DC) value, whereas, all other waveforms (u = 1,2,...,) give waveforms at increasing frequencies . These waveforms are called the *cosine basis function.* Note that these basis functions are orthogonal.

If the input sequence has more than *N nos of* sample points then it can be divided into sub-sequences of length *N* and DCT can be applied to these parts independently. Only the values of function will change in each sub-sequence. This is a important property, since it shows that the basis functions can be pre-calculated offline and then multiplied with the sub-sequences. This reduces the number of mathematical operations thereby providing computation efficiency.

4.1.2 The 2d Discreet Cosine Transform

The aim of the thesis is to study the efficiency of DCT on images. This requires the extension of ideas presented in the previous section to a 2-d space. The 2-D DCT is a direct extension of the 1-D DCT and is given by

$$C(u,v) = \frac{1}{4}\alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\frac{\cos(2x+1)u\pi}{2N}\frac{\cos(2y+1)v\pi}{2N} \quad (4)$$

for $u,v = 0,1,2,\ldots,N-1$. The inverse DCT transform is defined as

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\alpha(u)\alpha(v)C(u,v)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right] \quad (5)$$

for $x,y = 0,1,2,\ldots N-1$. The 2-D functions can be generated by multiplying the horizontally oriented 1-D functions with vertically oriented set of the same functions . The basis functions for $N = 8$ are shown. Again, it can be noted that the basis functions shows a progressive increase in frequency both in the vertical and horizontal direction. The top left basis function of results from multiplication of the DC component in with its transpose. Hence, this function assumes a constant value and is referred to as the DC coefficients.

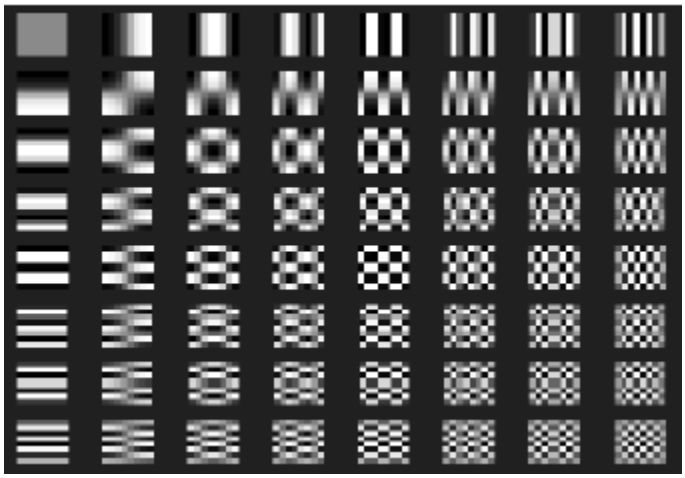


Fig 4.1. Two dimensional DCT functions for N = 8. Gray represents zero, white represents positive amplitudes, and black represents negative amplitude

4.2 Fundamental properties of Discreet Cosine Transform

This section provides some properties of the DCT which are of particular importance to image processing applications.

4.2.1)Decorrelation

The main advantage of image transformation is the removal of redundancy between neighboring pixels. So that uncorrelated transform coefficients which can be encoded independently. The normalized autocorrelation of the images before and after DCT is shown in Figure below. Clearly, the amplitude of the autocorrelation after the DCT operation is very small. Hence, it can be assumed that DCT exhibits excellent decorrelation properties.

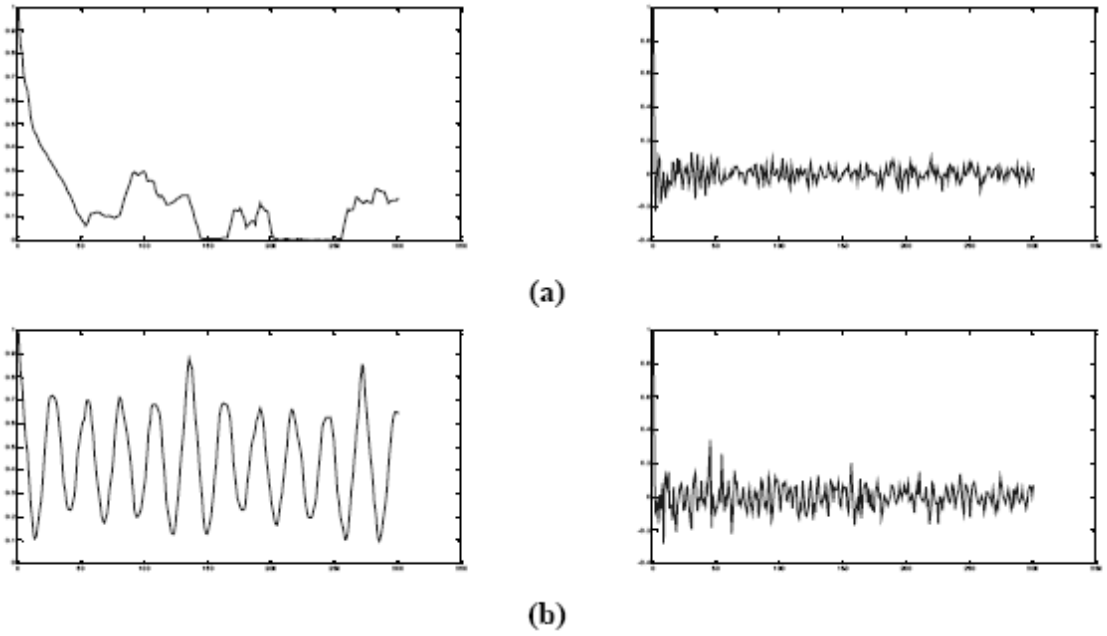


Fig 4.2. (a) Normalized autocorrelation of uncorrelated image before and after DCT; (b) Normalized autocorrelation of correlated image before and after DCT.

4.2.2)Energy Compaction

A transformation scheme should have the ability to pack input data into as few coefficients as possible. This allows the quantize to remove coefficients with relatively small amplitudes without visual distortion in the reconstructed image. DCT exhibits excellent energy compaction for high correlated images

(a) (b)

Fig 4.3 (a)correlated image, (b)uncorrelated image

Here we can see that the uncorrelated image has more sharp intensity variations than the correlated image. Therefore, the former has more high frequency content than the latter. Figure shows the DCT of both the images. So the uncorrelated image has its energy spread out, whereas the energy of the correlated image is packed into the low frequency region.

Hence, it can be inferred that DCT provides excellent energy compaction for correlated images. The energy compaction performance of DCT approaches optimality as image correlation approaches one .DCT provides optimal decorrelation for such images .

4.2.3)Separability

The DCT transform equation can be expressed as,

$$C(u,v) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right] \qquad (6)$$

for u,v = 0,1,2,...,N-1.

This property, known as *separability*, has the main advantage that *C (u,* v) can be computed in two steps by successive 1-Dimentional operations on rows and columns of an image. The arguments presented can be identically applied for the inverse DCT computation .For the hardware design, this property is utilized.

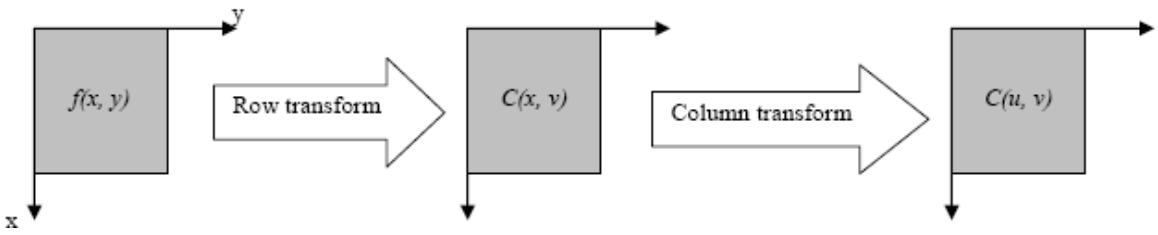


Fig 4.4. Computing of 2-D DCT using separability .

4) Symmetry

Here the row and column operations in Equation 6 reveal that those operations are functionally identical. Such a transformation is called a *symmetric transformation.* A separable and symmetric transform can be expressed in the form.

$$T = AfA \qquad (7)$$

Where *A* is an *n* x n symmetric transformation matrix .

$$\alpha(i, j) = \alpha(j)\sum_{j=0}^{N-1}\cos\left[\frac{\pi(2j+1)i}{2N}\right]$$

f is the *nxn* image matrix.

This is a useful property since it implies that the transformation matrix can be pre-computed offline and then applied to the image providing orders of magnitude improvement in computation efficiency.

4.2.5) Orthogonality

Assume that the inverse transformation of (7) as $f = A^{-1}TA^{-1}$.

Since DCT basis functions are orthogonal . So, the inverse transformation matrix of *A* is equal to its transpose i.e. $A^{-1} = A^{T}$. Therefore, and in addition to its decorrelation characteristics, this property provides some reduction in the pre-computation complexity.

Chapter 5

Different implementations of Discrete Cosine Transform

5.1 Different implementations of DCT

There are three different categories of approach for computation of the 2-D DCT. The first category of 2-D DCT implementation is indirect computation through other transforms like the Discrete Hartley Transform (DHT) and the Discrete Fourier Transform (DFT). The DHT-based algorithm has increased performance in throughput, latency, and turnaround time. A DFT calculates the odd-length DCT, which is not applicable to this project since the design must be compatible with JPEG standards.

In a different approach, the separability property of the DCT is used. An 8-point, 1-D DCT is applied to each of the 8 rows, and then again to each of the 8 columns. The 1-D algorithm that is applied to both the rows and columns is the same. So that, it is possible to use identical pieces of hardware to do the row computation and the column computation. A transpose matrix would separate the two as the functional description in figure 3.1 shows. So the fast algorithm for computing the 1-D DCT is usually selected. The high regularity of this method is very popular for reduced cell count and easy VLSI implementation.

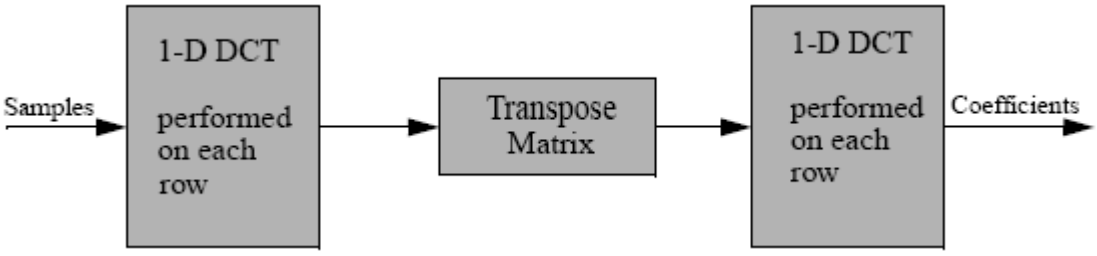


Figure 5.1:two dimensional DCT implementation

The third approach for computing the 2-D DCT is by a direct method using the results of a polynomial transform. Contrary to conventional method, where 16 1-D DCTs

used in the row-column decomposition, it uses all real arithmetic including 8 1-D DCTs, and stages of pre-adds and post-adds (a total of 234 additions) to compute the 2-D DCT. So, the number of multiplications for most implementations should be halved as multiplication appears within the 1-D DCT.

Since row-column decomposition is very useful for VLSI implementation, that implementation is considered in this project. In that implementation, starts with one-dimensional transform.

1-D algorithm	Multiplications	Additions	Compensation
Theoretical Equations (conventional)	64	56	Not applicable
Chen's algorithm	32	32	Not Applicable
CORDIC algorithm	0	12	Depends on angle
New CORDIC algorithm by Zhang	0	12	No compensation

Table 5.1:Comparison between algorithms in terms of Multiplications and additions

The final scale factors of computation on each row cannot be accommodated into the quantization table because the scale factors are distinct for every coefficient. if one optimizes the 2-D DCT computation by incorporating required multiplications into the quantization matrix, the design no longer calculates the DCT. It computes a version in which each coefficient needs to be scaled appropriately and is dependant on the presence of a quantization table. So, this 1-D DCT algorithm is optimized for use only within a compression core, such as JPEG. Since it is the intent of the project to have a stand alone DCT core, this approach is not feasible.

While the direct method of 2-D DCT calculation claims to decrease the number of multiplications in the row-column approach by a factor of two , is not always true. This is because the direct method must do some post processing after the 1-D DCT computation stage so the constant scale factors cannot be incorporated into the quantization matrix.

The theoretical implementation of 1-D DCT algorithm is a simple transform given by

Y=AX, where X is 1-Dimentional array of data.

Where it involves 64 multiplications and 56 additions to compute the entire the 1-D DCT and also it is very complex to route over FPGA. The matrix A is given by

$$A = \begin{bmatrix} d & d & d & d & d & d & d & d \\ a & c & e & g & -g & -e & -c & -a \\ b & f & -f & -b & -b & -f & f & b \\ c & -g & -a & -e & e & a & g & -c \\ d & -d & -d & d & d & -d & -d & d \\ e & -a & g & c & -c & -g & a & -e \\ f & -b & b & -f & -f & b & -b & f \\ g & -e & c & -a & a & -c & e & -g \end{bmatrix}$$

5.1.1Chen's algorithm

The fast 1-D DCT algorithm that was selected for use in both the direct and row-column 2-D approaches was developed by Chen .The 8-point, 1-D DCT, written in matrix factorization.

$$\begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \end{bmatrix} = \frac{1}{2}\begin{pmatrix} A & A & A & A \\ B & C & -C & -B \\ A & -A & -A & A \\ C & -B & B & -C \end{pmatrix}\begin{bmatrix} x_0 + x_7 \\ x_1 + x_6 \\ x_2 + x_5 \\ x_3 + x_4 \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ x_3 \\ x_5 \\ x_7 \end{bmatrix} = \frac{1}{2}\begin{pmatrix} D & E & F & G \\ E & -G & -D & -F \\ F & -D & G & E \\ G & -F & E & -D \end{pmatrix}\begin{bmatrix} x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{bmatrix} \qquad (1)$$

Where A=cos (pi/4), B=cos (pi/8), C=sin (pi/8), D=cos (pi/16), E=cos (3*pi/16), F=sin (3*pi/16), G=sin (pi/16)

As mentioned earlier the 1-D DCT can be expressed

as $$F(u) = \frac{1}{2}c(u)\sum_{y=0}^{7} f(x)\frac{\cos(2x+1)u\pi}{16} \qquad (2)$$

where c(u)= $\frac{1}{\sqrt{2}}$

5.1.2 DCT using CORDIC architectures.

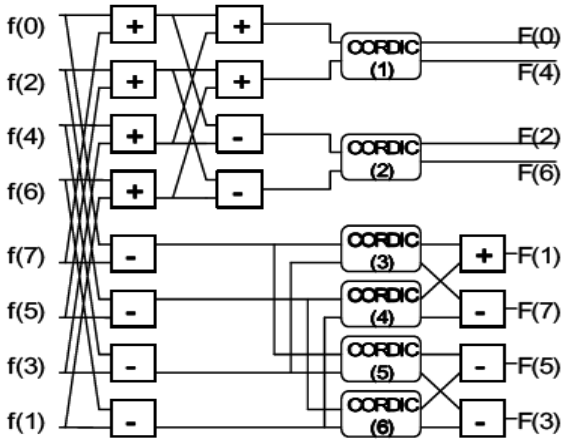


Fig5.2:1-D DCT architecture using CORDIC algorithm

$$F(0) = \{f(0) + f(7) + f(3) + f(4)\}\cos\left(\frac{\pi}{4}\right) + \{f(1) + f(6) + f(2) + f(5)\}\sin\left(\frac{\pi}{4}\right)$$
$$F(4) = \{f(0) + f(7) + f(3) + f(4)\}\cos\left(\frac{\pi}{4}\right) + \{f(1) + f(6) + f(2) + f(5)\}\sin\left(\frac{\pi}{4}\right) \quad (3,4)$$

Therefore, in order to compute both F (0) and F (4), we need one CORDIC processor. F (2) and F (6) can be obtained by using the rotation mode of CORDIC. For F (1) and F(7), F(5), and F(3), we need four CORDIC processors. So we can use six CORDIC processors for the 2D-DCT by applying the 1DDCT two times.

The number of iterations can be decreased, since the coefficients for 8×1 DCT are fixed. The compensation process for the final CORDIC calculation can be composed of adder and shifter without multiplier as expressed below.

$$X_{i+1} = X_i(1 + \gamma_i.F_i)$$
$$Y_{i+1} = Y_i(1 + \gamma_i.F_i) \quad (5)$$

where γ_i has the value ± 1 and F_i is for shift operation.

Table 5.2 shows the detailed number of rotation for iterations and compensation in six CORDIC processors.

processor	CORDIC(1)	CORDIC(2)	CORDIC(3)(6)	CORDIC(4)(5)
angle	$\frac{\pi}{4}$	$\frac{3\pi}{8}$	$\frac{7\pi}{16}$	$\frac{3\pi}{16}$
CORDIC iteration [σ ,i]				
1	-1,0	1,0	1,0	1,1
2		1,2	1,1	1,3
3		1,3	1,3	1,10
4		1,6	1,10	1,14
5		1,7		
	Compensation iterations $[1+\gamma_i(i)F_i(i)]$			
1	$1-\frac{1}{4}$	$\frac{1}{2}+\frac{1}{8}+\frac{1}{64}$	$\frac{1}{2}+\frac{1}{8}$	$1-\frac{1}{8}$
2	$1-\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{8}+1$	$1+\frac{1}{64}$
3	$1+\frac{1}{256}$		$1+\frac{1}{4096}$	$1+\frac{1}{1024}$
4	$1+\frac{1}{512}$			$1+\frac{1}{4096}$
5	$1+\frac{1}{4096}$			

Table 5.2: number of rotation for iterations and compensation

Due to the disadvantages of the conventional CORDIC algorithm like fixed number of iterations and the effect of the scaling factor, the conventional CORDIC algorithm lost its importance. But here we can see that new CORDIC algorithm is not so efficient with respect to conventional CORDIC algorithm. The architecture for implementing 2D DCT using both CORDICs is same. Now we'll see the algorithm for the new Cordic algorithm is given in table 5.3

processor	CORDIC(1)	CORDIC(2)	CORDIC(3)(6)	CORDIC(4)(5)
angle	$\frac{\pi}{4}$	$\frac{3\pi}{8}$	$\frac{7\pi}{16}$	$\frac{3\pi}{16}$
CORDIC iteration[σ ,i]				
1	[1,2]	[1,2]	[1,3]	[1,2]
2	[1,2]	[1,3]	[1,4]	[1,2]
3	[1,2]	[1,6]	[1,7]	[1,4]
4	[1,5]	[1,9]	[1,10]	[1,6]
5	[1,9]	[1,13]		[1,7]
6	[1,10]			[1,9]
7				[1,10]

Table 5.3: Algorithm of the new Cordic algorithm used for the Calculation of 1-D DCT.

Chapter 6

Design of CORDIC and DCT Core

6.1 DESIGN OF CORDIC MODULE

CORDIC module are built with three fundamental blocks.

---Pre Processor

---Post Processor

---CORDIC core

The core is built using pipeline stages, each stage representing a single step in iteration process. The processor input takes x and y coordinates of a given vector as signed values, the processor core rotates the original vector to align it with x-axis .After the input data processing the calculated data of rotational angle and vector magnitude is provided in processor output.

The processor input has 16 bit input data width,20 bit output data width and 15 nos of iteration is done.

6.1.1 PRE-PROCESSOR

To fit to the range allowable for core processing, that requires a pre-processor which detects the right quadrant where the given vector is located and fits it into the range from 0 to 45 degrees.

6.1.2 POST-PROCESSOR

After being processed in the core ,the results need to be corrected which is called vector length correction done by post-processor by multiplying its value by 0.85879,the angle correction done by rotating the angle to the corr. Quadrant of the plane.

6.1.3CORDIC CORE

CORDIC core performs the cordic algorithm on the input data sent by the pre-processor. It contains various pipelined stages for better performance. The stage performs single iteration step. It contains arc tan table for each iteration and the logic for handling the x,y,z values.

Top level schematic of CORDIC module

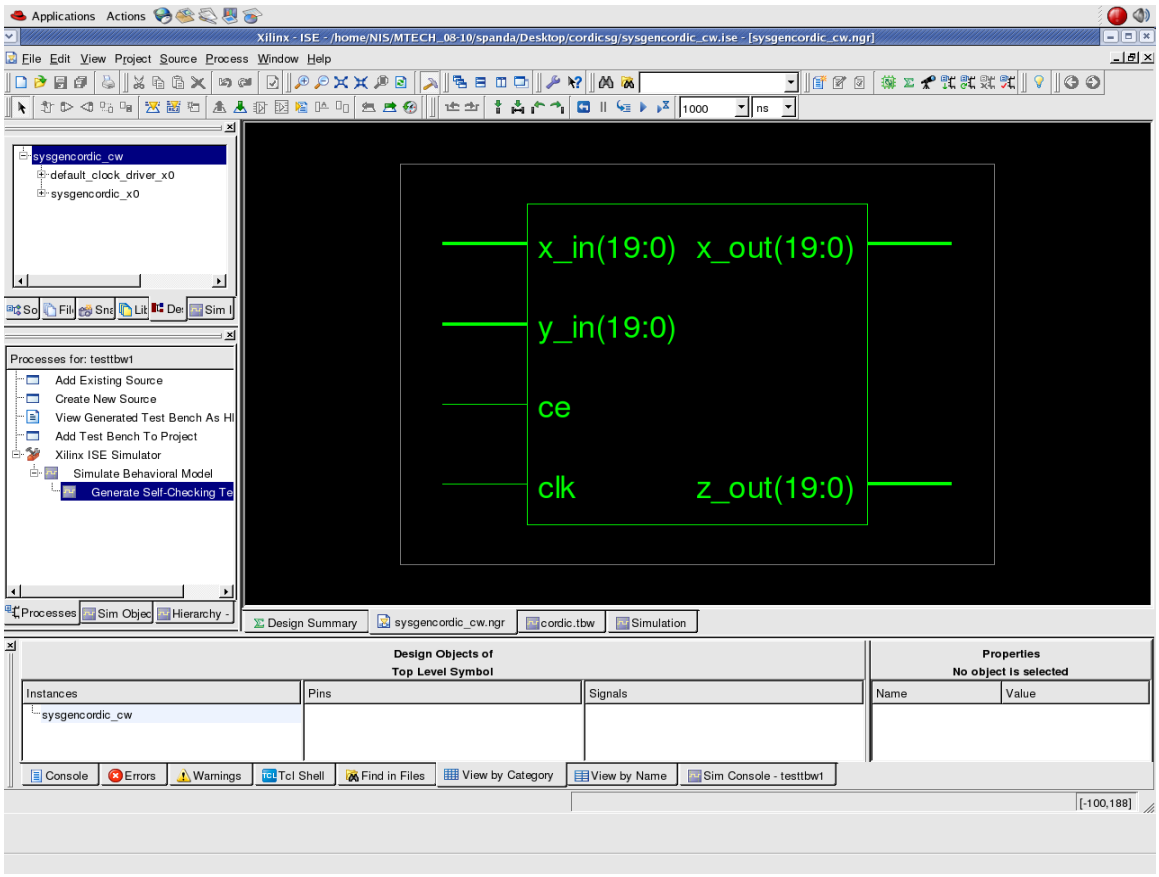


Fig6.1 **Top level schematic of CORDIC module**

CORDIC POCESSOR WITH ALL SUB_MODULES

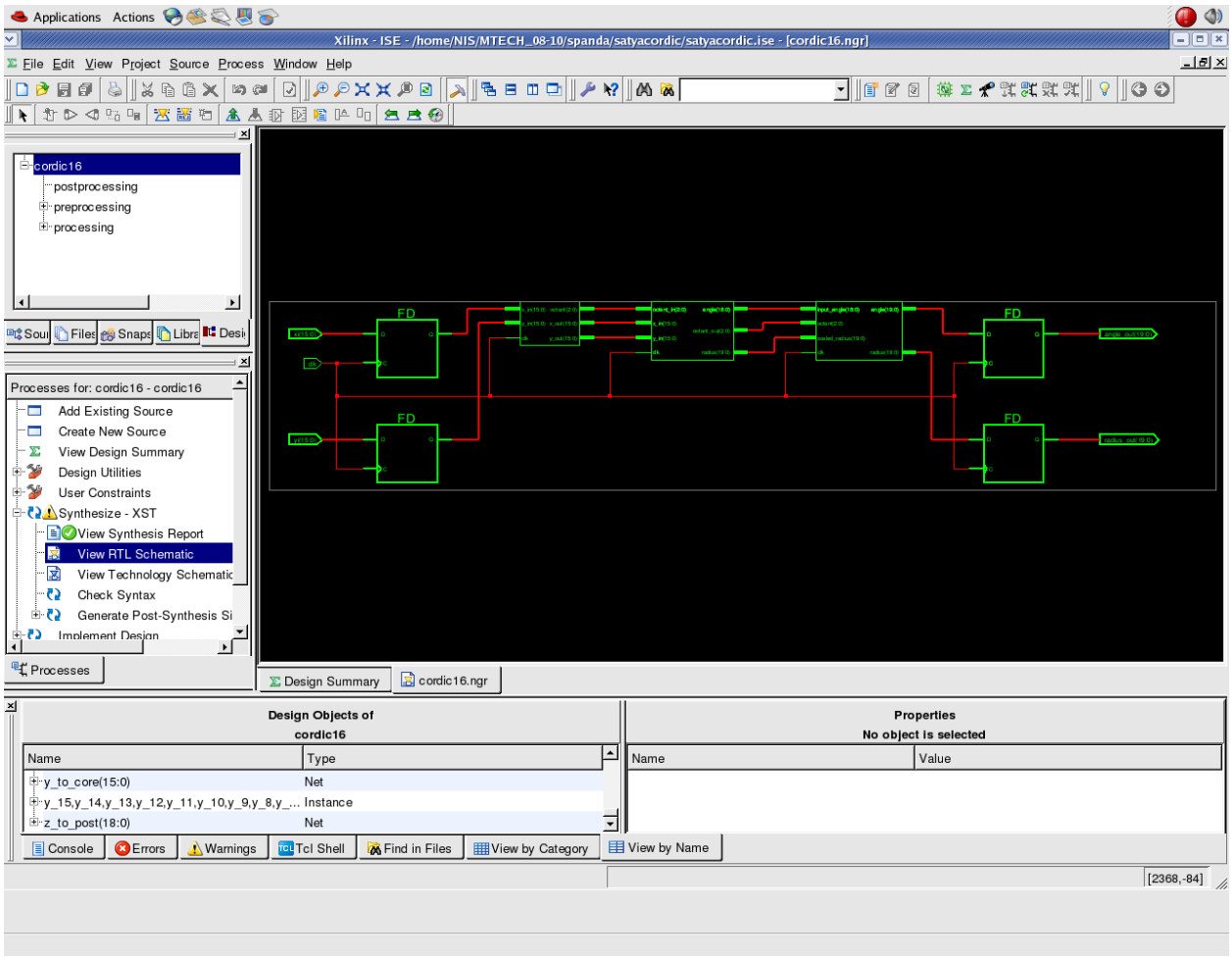


Fig 6.2 **CORDIC POCESSOR WITH ALL SUB_MODULES**

DIFFERENT PIPELINED STAGES INSIDE CORDIC CORE

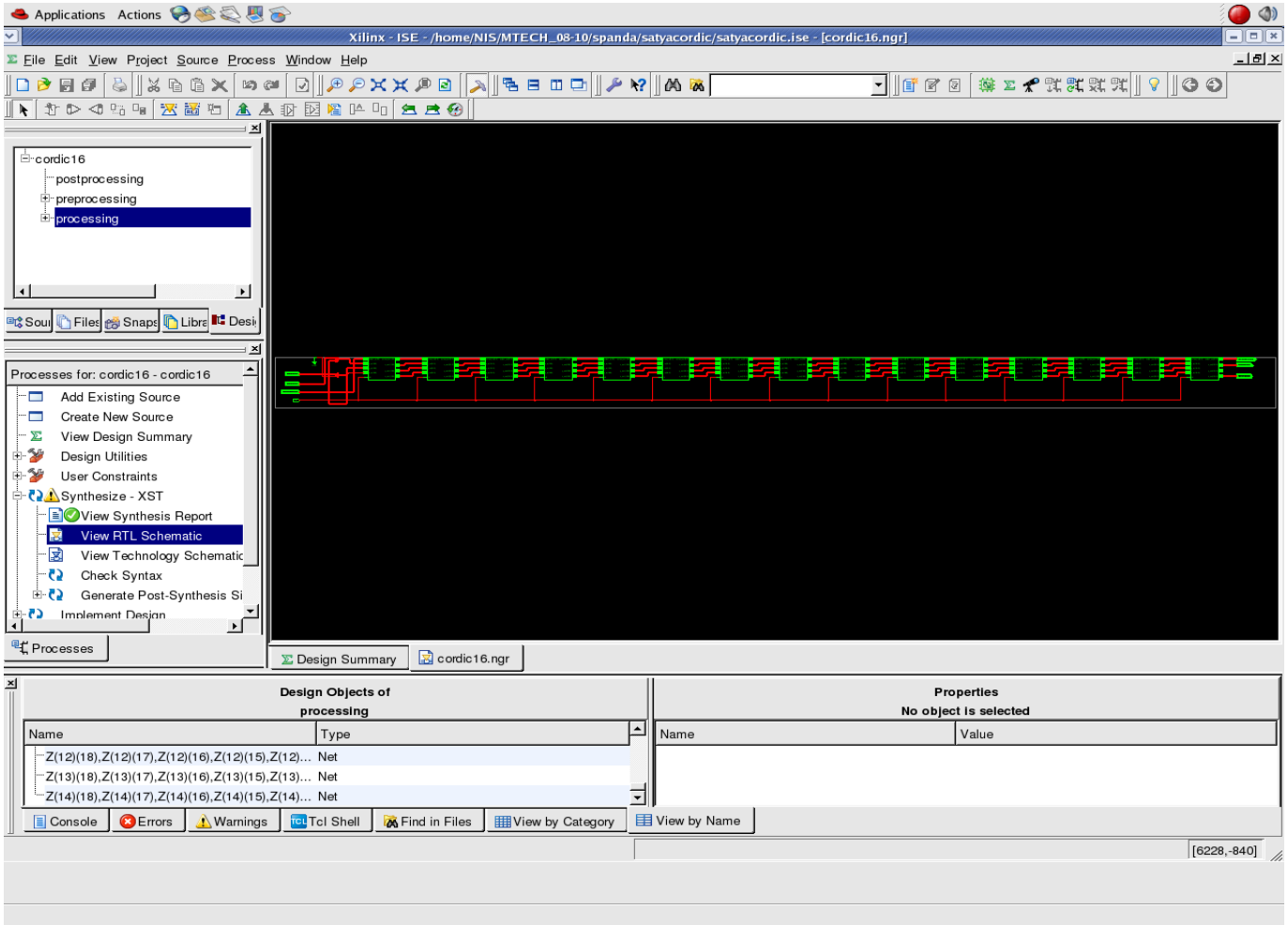


Fig6.3 **DIFFERENT pipelined stages with CORDIC core**

SINGLE STAGE IMPLEMENTATION OF CORDIC ALGORITHM

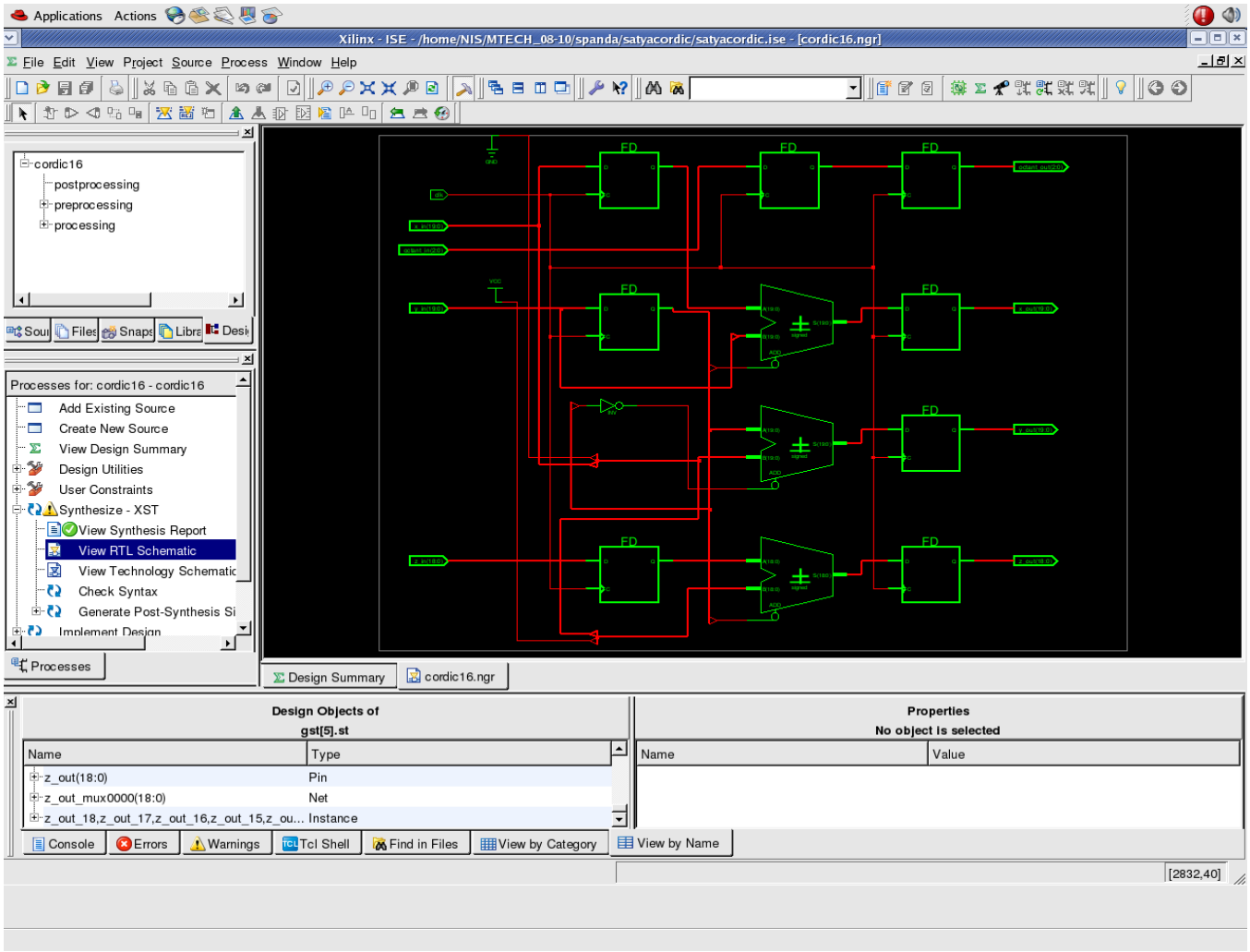


Fig 6.4 Single stage implementation of CORDIC algorithm

DESIGN OF CORDIC MODULE USING MAT LAB SIMULINK

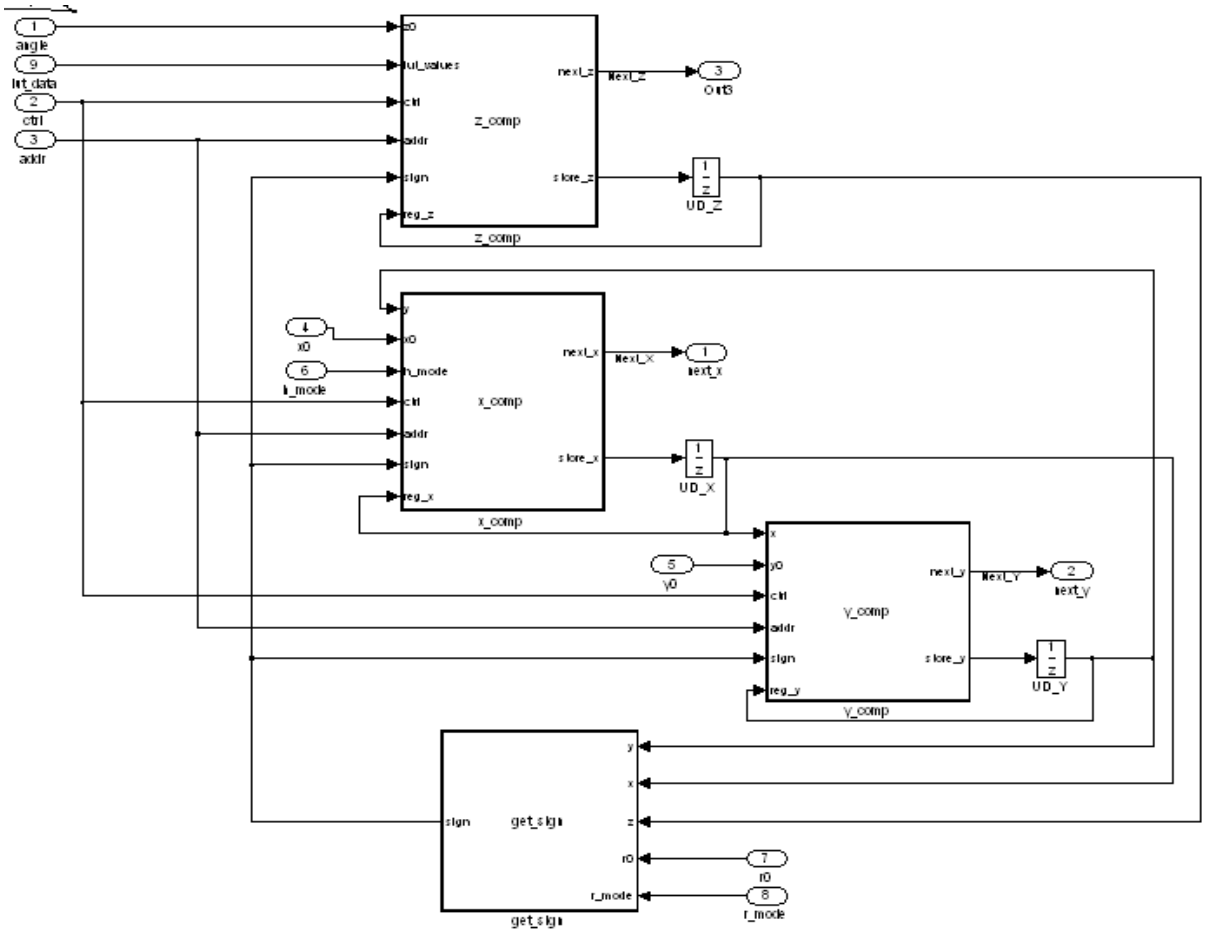


Fig 6.5 DESIGN OF CORDIC MODULE USING MAT LAB SIMULINK

System generator modeling of the above simulink model

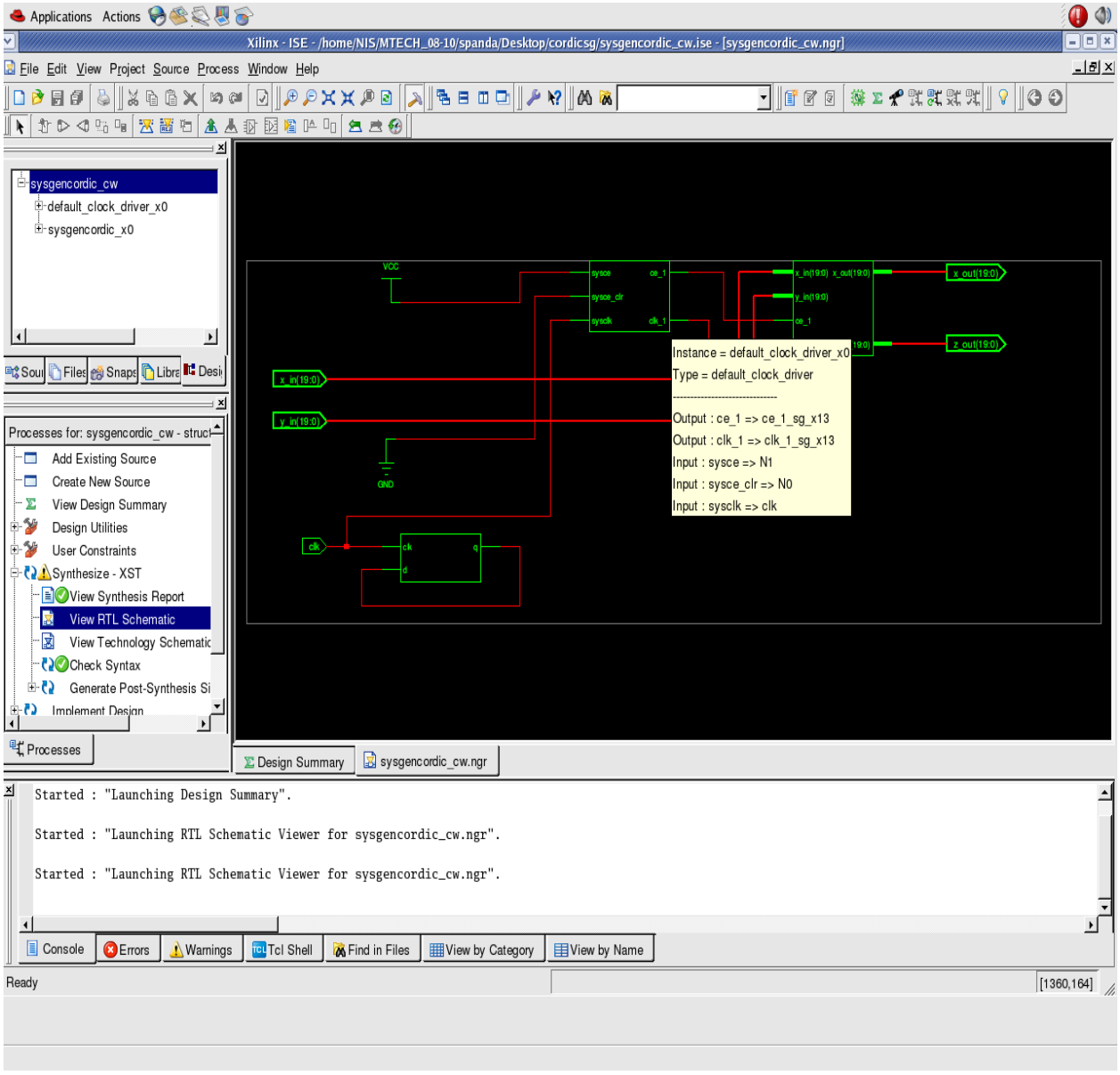


Fig 6.6 System generator modeling of the above simulink model

6.2 DESIGN OF DCT CORE

The design flow of DCT core is shown particularly in the following flow-chart given below…

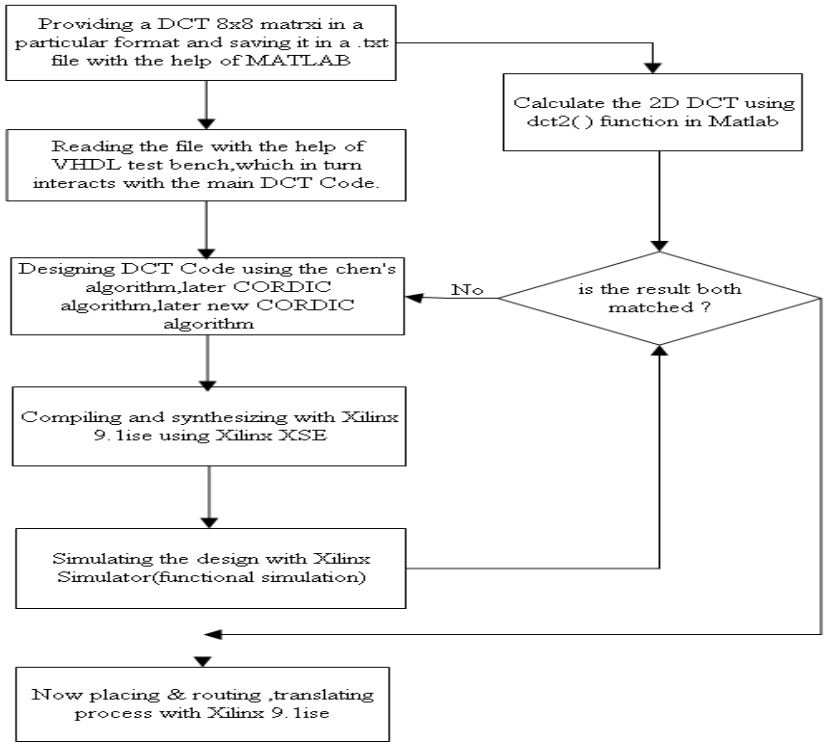


Fig 6.6: Flow chart approach in the DCT design

The format of input word is different for different implementations where for example take the case of implementation of Chen's algorithm where the input word length is 8 bits. But in the case of implementation of CORDIC implementation, the input word is taken in the form 1QN format which if a fixed format. For that purpose all the data must be less than 1.So whatever is the data we have to make it less than 1 for Conventional CORDIC algorithm, where as for new CORDIC algorithm we have to

make data in the form of 32 bits. CORDIC, as virtually any FPGA DSP core does, utilizes fixed-point arithmetic. In particular, the numbers the core operates with are presented as two's complement signed fractional numbers. To identify the position of a binary point separating the integer and fractional portions of the number, the Q format is commonly used. An mQn format number is an (n + 1)-bit signed two's complement fixed-point number: a sign bit followed by n significant bits with the binary point placed immediately to the right of the m most significant bits. The m MSBs represents the integer part, and (n-m) LSBs represent the fractional part of the number, called the mantissa. Table 6.1 depicts an example of a 1Qn format number.

Bit 2^n	Bit 2^{n-1}	Position of the Binary Point	Bits $[2^{n-2} : 2^0]$
Sign	Integer bit		Mantissa

Bit 2^n	Position of the Binary Point	Bits $[2^{n-1} : 2^0]$
Sign		Mantissa

Table 6.1: QN Format Number.

6.2.1 I/O linear Format:

The CORDIC engine utilizes the 1Qn format shown in Table 3. Though the 1Qn format numbers are capable of expressing fixed-point numbers in the range from (-2^n) to ($2^n - 2^{m+n}$), the input linear data must be limited to fit the smaller range from (-2^{n-1}) to (2^{n-1}). In terms of floating-point numbers, the input must fit the range from -1.0 to1 .0. For example, the 1Q9 format input data range is limited by the following 10-bit numbers:
Max input negative number of -1.0:

1100000000 ⇔ 11.00000000

Max input positive number of +1.0:

0100000000 ⇔ 01.00000000

This precaution is taken to prevent the data overflow that otherwise could occur as a result of the CORDIC inherent processing gain. The output data obviously do

not have to fit the limited range. To convert floating-point linear input data to the 1Qn format, follow the simple rule in EQ 10:

$$1Qn\ Fixed\text{-}Point\ Data = 2^{n-1} \times Floating\text{-}Point\ Data \quad (1)$$

Here it is assumed the floating-point data are presented in the range from -1.0 to 1.0. The product on the right-hand side of Eq (1) contains integer and fractional parts. The fractional part has to be truncated or rounded. shows a few examples of converting the floating-point numbers to the 1Q15 format.

To convert the 1Qn format back to the floating-point format, use EQ 11.

$$Floating\text{-}Point\ Data = 1Qn\ Fixed\text{-}Point\ Data/2^{n-1} \quad (2)$$

Suppose we are using the input in which one of the number is 34,then we have to convert to 1Q17 format where we have to divide 34 with 1000 and then we have to multiply with 2^16 and then round it…so that the end result is round(0.034*2^16) which results in 2228.Similarly the angle format for the CORDIC design is also the same, as mentioned above.

For the Design of 2D DCT using the New Cordic Algorithm, all the inputs must be 32 bits wide. All inputs are converted from decimal to fixed-point binary representation in Matlab. For this 32-bit design, the least 31 bits are used to represent the decimal fraction. The Most Significant Bit (MSB) is used as the sign bit. To check the output data of x' and y' at each rotation angle, we first convert angle θ from binary to decimal representation, and then divided by 2^P, where P is the number of bits used to represent the decimal fraction. Then we can calculate the value of cos and sin θ for the estimation of output x' and y' respectively. If x' and y' are almost the same as the sine and Cosine value that we calculate, then we can say the operation of the CORDIC ip is Correct. For example

$$(\theta)_2 = (00000010001110111110100011010100)_2 \ \textit{radians}$$
$$(\theta)_{10} = (37480660)_{10} / 2^p \ \textit{radians}$$
$$= (37480660)_{10} / 2^{31} = 0.0175 \ \textit{radians}$$

$$\cos(\theta) = \cos(0.0175) = 0.9998$$
$$(x')_2 = (01111111111110110011011100000100)_2$$
$$\therefore (x')_{10} = (2147200000)_{10} / 2^p = (2147200000)_{10} / 2^{31} = 0.9999 \approx \cos(\theta)$$

$$\sin(\theta) = \sin(0.0175) = 0.0175$$
$$(y')_2 = (00000100010111111111110001011100)_2$$
$$(y')_{10} = (36699228)_{10} / 2^p = (36699228)_{10} / 2^{31} = 0.0171 \approx \sin\theta$$

The above example verifies that output x' and y' are correct for its given rotation angle. Several outputs with different input rotation angles are chosen randomly to verify the correctness by following the steps illustrated in the example. Moreover, the simulation results are generated in waveforms so that we check not only the value of the outputs but also see if there is any timing matching problem within the overall design.

In the case of the Design of 2D DCT using the new CORDIC algorithm is shown below.

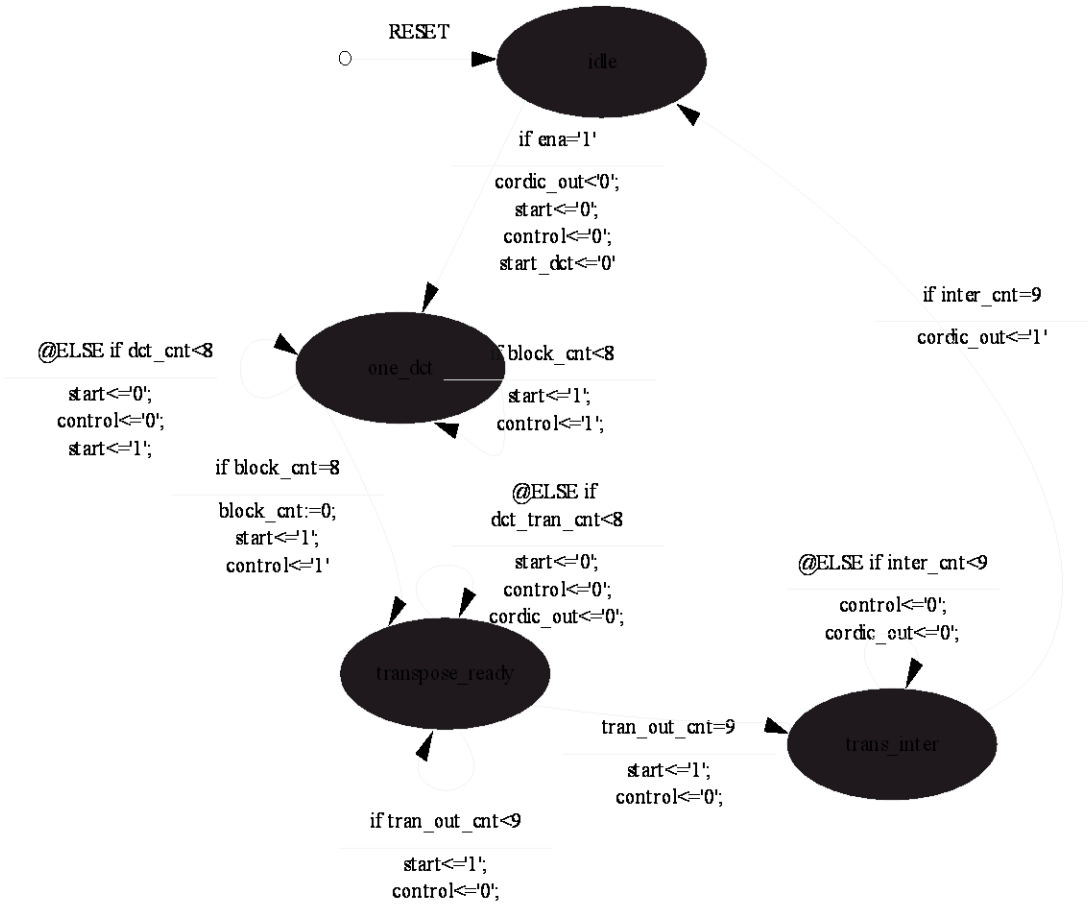


Fig 6.7 : FSM for DCT Design using CORDIC algorithm

In the design of DCT core using CORDIC algorithm, algorithms AR Cordic as well as the new CORDIC algorithm, the same state diagram is used. From the above state diagram it is implied the complexity of DCT architecture is less in the case of CORDIC architecture, where in the conventional one consists of multiplications, where in the normal CORDIC architecture consists of only rotations and shiftings.

Now consider about the design of Matrix transposer cell which is necessary a some sort of transpose buffer. The need for real-time implementation of the transposition operation is felt particularly in image processing applications as they are dominated by matrix based techniques. For example, a wavelet operation on a two-dimensional array of data is executed as follows: First, the wavelet operation is executed on the rows (columns) of data followed by a transposition operation. This process is then repeated on the columns (rows) of data.

The external structure of DCT module is given below:

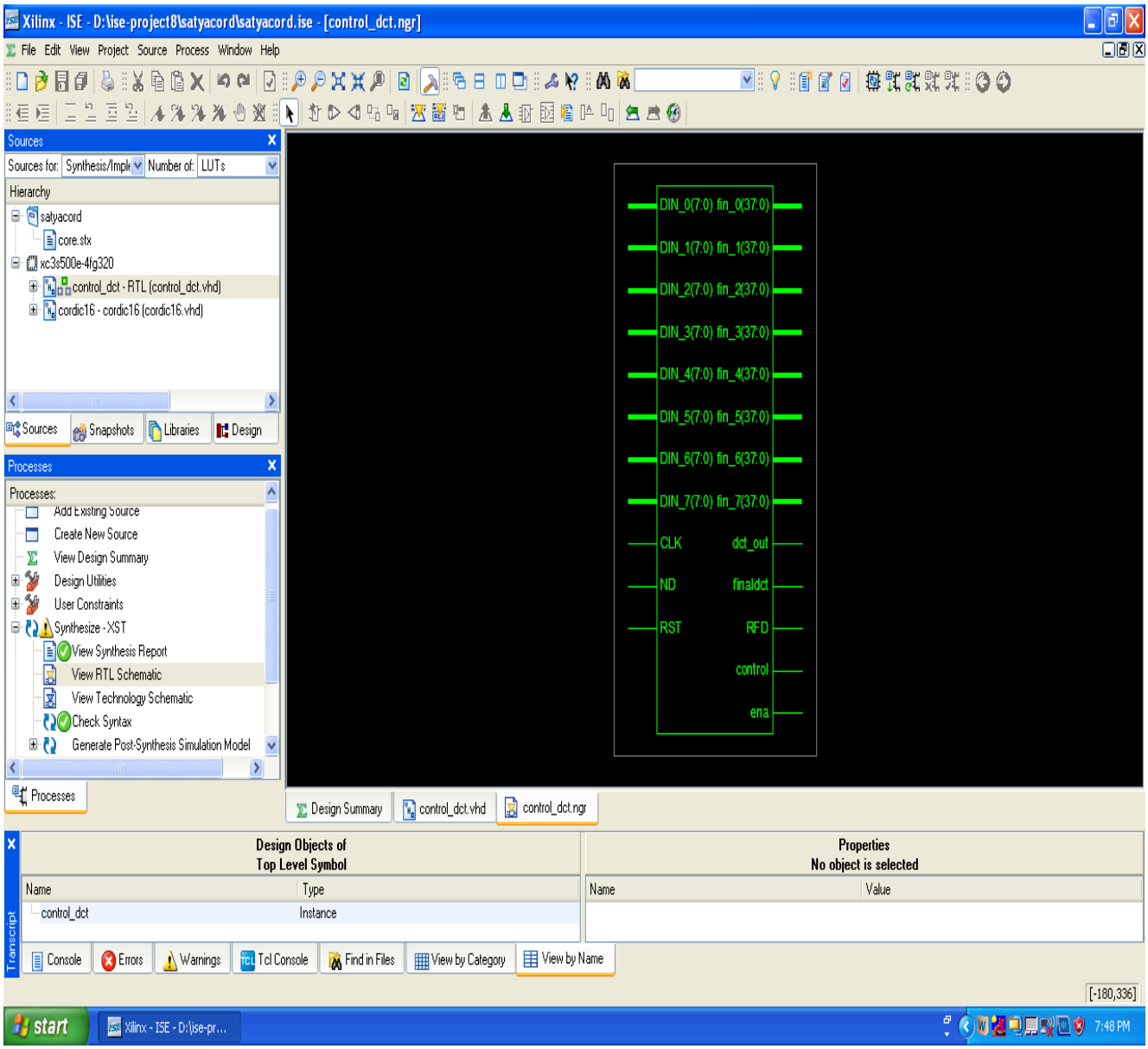


Fig 6.8 Top level schematic of DCT module

DIN_0…6(DATA INPUTS):

Din_0,Din_1,Din_2,Din_3,Din_4,Din_5,Din_6,Din_7 are inputs of 8 bits width .The Module will read the data when ND is high.

ND(NEW DATA):

When this input signal is high it indicates that valid data is available at the input DIN. If RFD is high then the module reads this data.

RST (Reset):

Reset allows user to restart the 2-D DCT process.

CLK (Clock):

This clock signal is used to synchronize the module and data input output operations

DOUT_0..6 (Data Output):

This output ports provides the results of 2-D DCT. When control signal finaldct is high,the DOUT_0,DOUT_1, DOUT_2, DOUT_3,DOUT_4,DOUT_5,DOUT_6,DOUT_7 is valid. The bit width of the outputs is 38.

FINAL DCT:

This signal indicates that whether the data at the output port is valid or not.

RTL SCHEMATIC OF THE ABOVE DCT MODULE

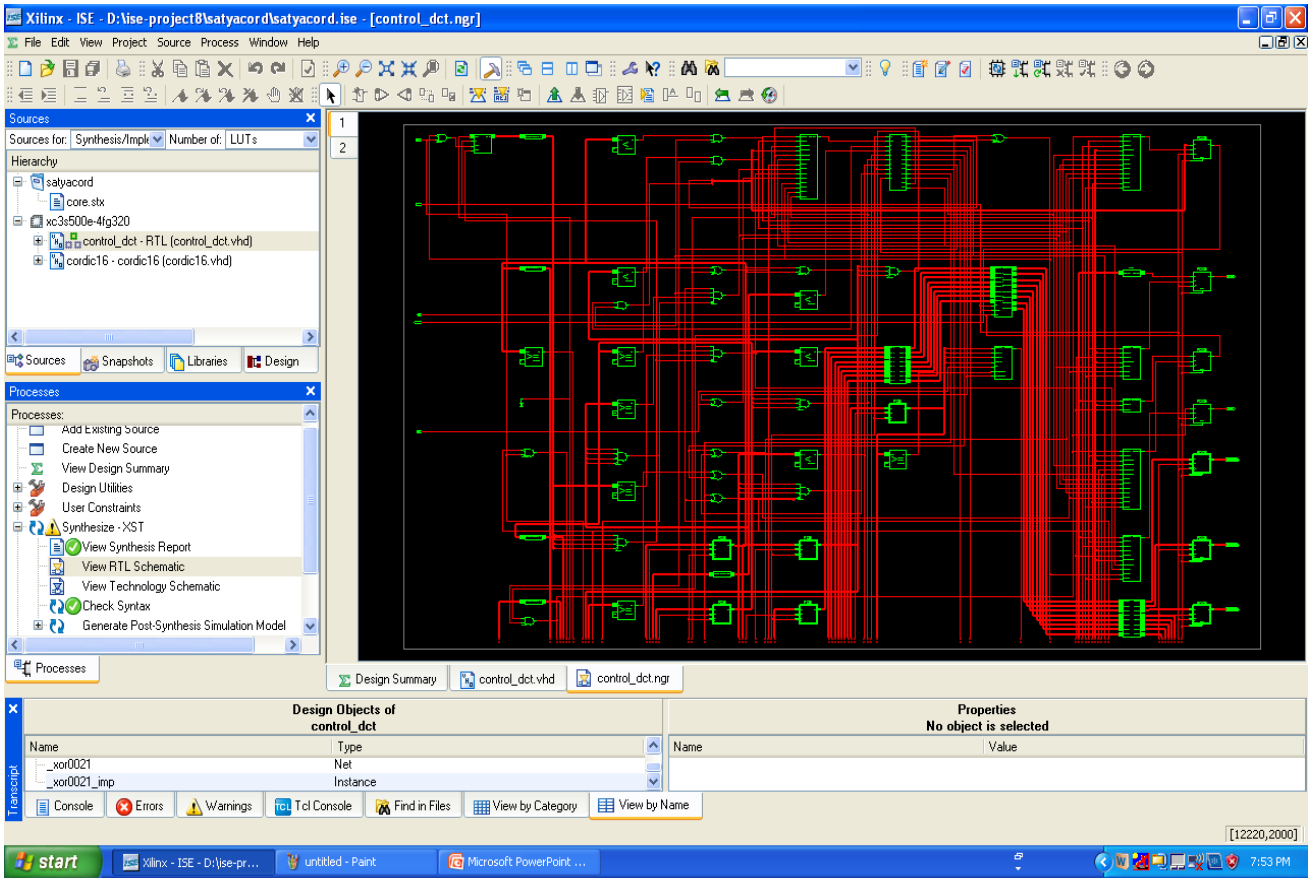


Fig6.9 RTL SCHEMATIC OF THE DCT MODULE

Chapter 7

Simulation Results

7.1 Simulation results of CORDIC

At first the designed CORDIC module is tested with 16 bit data to get the underlying simulation.

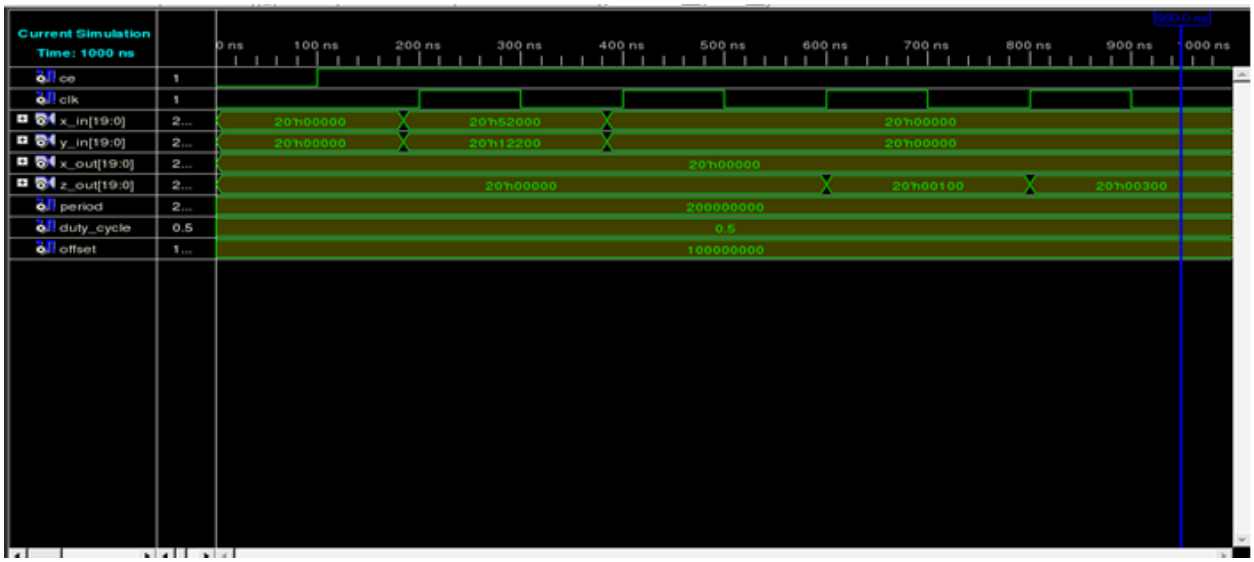


Fig 7.1 Simulation of CORDIC module

Then the system generator model of CORDIC designed with mat lab is tested with the same data.

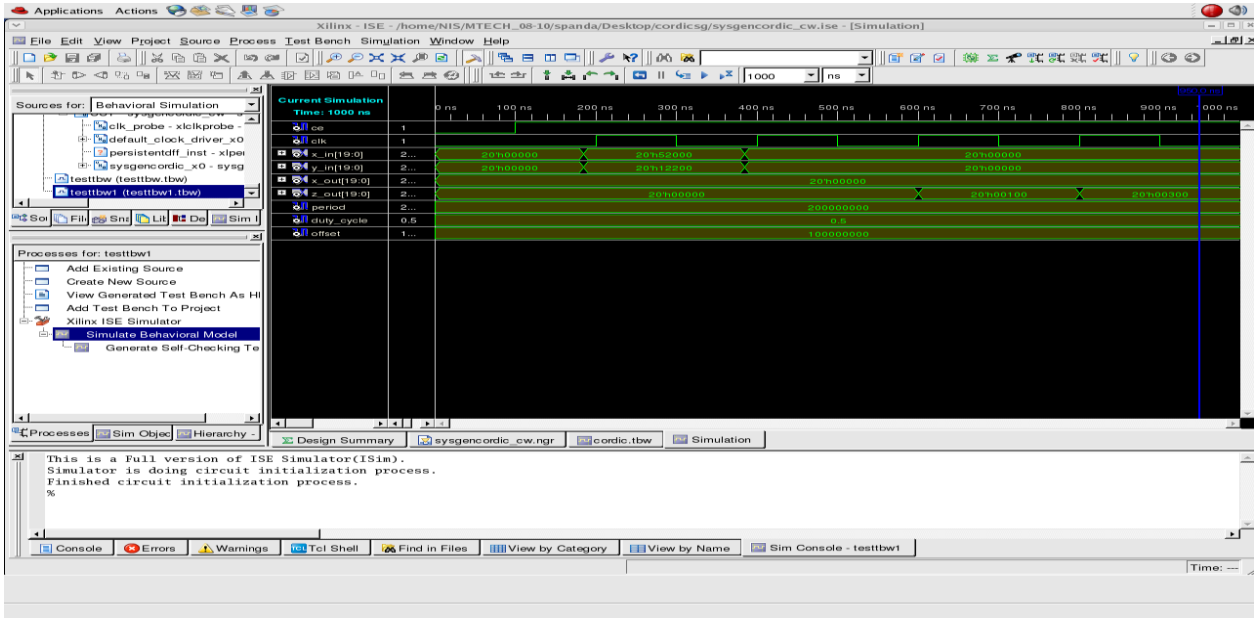


Fig 7.2 Simulation of System generated module

From the above two simulation comparision ,we can assume that our design is according to the specification.

7.2 SIMULATION OF DCT CORE

The inputs to the DCT core using CORDIC algorithm is given from a text file which contains the data as

23 44 34 33 34 29 35 23
22 54 54 33 34 29 35 23
22 64 64 33 34 29 35 33
11 44 64 33 34 29 35 43
45 64 84 33 34 29 35 53
39 76 60 27 33 31 31 52
23 72 85 30 32 34 31 68
41 31 77 79 30 31 28 39

Actually this data is the starting 8x8 matrix in a file .Now we did the 2D DCT using matlab and got the result as

Y= [159.5000 2.7683 4.304 -0.2992 0.2400 -0.539 -4.5294 5.6385
7.9473 -0.779 0.547 -4.9323 1.9602 2.9784 -3.7971 3.3222
5.349 -0.274 -15518 1.7250 -0.6765 -0.4525 1.8499 -2.2002
1.2619 1.390 1.7039 0.942 -0.706 -1.3686 0.2143 1.1422
-1.4000 -1.497 -0.3431 -1.596 1.700 1.189 -1.4815 -0.6729
.2107 0.3561 -1.821 -0.1061 -2.0116 0.097 1.6866 0.7552
-2.7028 0.3650 3.0999 1.6355 1.6332 -2.2546 -1.182 -0.911
1.2259 -0.3152 -2.326 -1.5945 -0.8846 1.9693 0.532 0.6540]

Now the same file is compiled, synthesized and simulated using Xilinx9.1ise and directly from the Xilinx itself we are saving the result in a file

Simulation result of DCT core in Xilinx

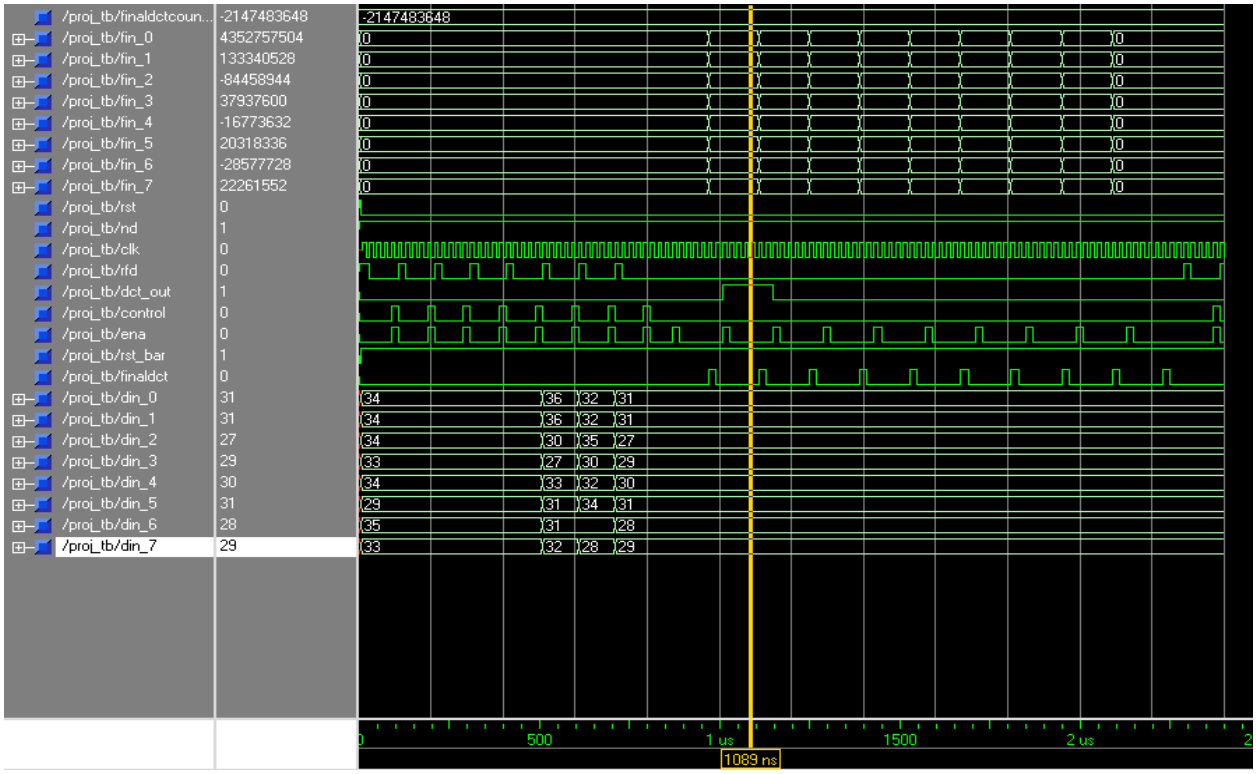


Fig 7.3 Simulation of DCT module

Y1= [159.5464 2.7684 4.307 -0.2995 0.2400 -0.539 -4.5294 5.6385
7.9373 -0.779 0.547 -4.9323 1.9602 2.9784 -3.7971 3.3222
5.349 -0.274 -15518 1.7250 -0.6765 -0.4525 1.8499 -2.2002
1.2619 1.390 1.7039 0.942 -0.706 -1.3686 0.2143 1.1422
-1.4000 -1.497 -0.3431 -1.596 1.700 1.189 -1.4815 -0.6729
.2107 0.3561 -1.821 -0.1061 -2.0116 0.097 1.6866 0.7552
-2.7028 0.3650 3.0999 1.6355 1.6332 -2.2546 -1.182 -0.911
1.2259 -0.3152 -2.326 -1.5945 -0.8846 1.9693 0.532 0.6540]

So there is 1% error which can easily neglected. So with the help of the CORDIC algorithm even though there is an error, we can compensate this with low power and less area as well more compact design. This result is shown in the data results soon. Now considering the design with DCT using the New CORDIC algorithm. Since in the design in order to maintain a good precision, we have to take a more bit width.

Now consider the synthesis reports created by Xilinx 9.1ise.The family used for synthesizing are tabled below.

Device Family	Virtex2P
Device	XC2VP70
Package	FF1704
Speed	-6

Table 8.1 synthesis report created by xilinx

Synthesis results show that the DCT using Chen's algorithm takes a lot of area, consume much power as there is a multiplier factor present in the algorithm, DCT using the new CORDIC algorithm is also inferior compared to the Angle Recoded Cordic algorithm in terms of area and power consumption. Also there must be trade off between required precision and the input bit width.

Chapter 8

CONCLUSION

8.1 CONCLUSION

The CORDIC algorithm is a powerful and widely used tool for digital signal processing applications and can be implemented using PDPs (Programmable Digital Processors). But a large amount of data processing is required because of complex computations. This affects the cost, speed and flexibility of the DSP systems. So, the implementation of DFT using CORDIC algorithm on FPGA is the need of the day as the FPGAs can give enhanced speed at low cost with a lot of flexibility. This is due to the fact that the hardware implementation of a lot of multipliers can be done on FPGA which are limited in case of PDPs.

In this thesis the CORDIC module is simulated using Xilinx which is then used for simulation of Discrete Cosine Transform. Then the implementation of DCT processor using CORDIC module is done on XILINX . The results are verified by test bench generated by XILINX simulator. This thesis shows that CORDIC is available for use in DSP Processors based computing machines, which are the likely basis for the next generation DSP systems. It can be concluded that the designed RTL model for CORDIC and DCT function is accurate and can work for real time applications.

8.2 Future Scope of work

The future scope should include the following

- Implementation of CORDIC algorithm for higher order DCT systems
- DCT computation and simulation for more number of points
- Implementation and simulation for DHT and DST calculations

REFERENCES

.

[1] Javier Valls, Martin Kuhlmann, and Keshar K. Parhi."Evaluation of CORDIC algorithms for FPGA design". Journal of vlsi signal processing.vol.32,2008

[2]Maharatna, K., Troya, A., Krstic, M., Grass, E., and Jagdhold, U.: 'A CORDIC like processor for computation of arctangent and absolute magnitude of a vector'. Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS), 2007

[3]Maharatna, K., Dhar, A.S., and Banerjee, S.: 'A VLSI array architecture for realization of DFT, DHT, DCT and DST', Signal Process., 2004

[4]Deprettere, E., and Udo, R.: 'The pipelined CORDIC'. Internal Report Network Theory Section, Delft University of Technology, 2003.

[5] R. C. Conzalez and R. E. Woods, *Digial Image Processing*,Prentice-Hall, Inc., New Jersey, 2001.

[6] Iain E. G. Richardson, *Video Codec Design*, JohnWiley & Sons Ltd, Atrium, England, 2002.

[7] J. Li and Shih Lien Lu, "Low Power Design of Two- Dimensional DCT," in *IEEE Conf. on ASIC and Exhibit*,Sept. 1996, pp. 309–312.

[8] S.F. Hsiao, Y.H. Hu, T.B. Juang, and C.H. Lee, "Effi-cient VLSI Implementations of Fast Multiplier less Approximated DCT Using Parameterized Hardware Modules for Silicon Intellectual Property Design," *IEEE Trans. Circuits Syst. I*, vol. 52, pp. 1568–1579, Aug.2005.

[9] N. J. August and Dong Sam Ha, "Low Power Design of DCT and IDCT for Low Bit Rate Video Codecs," *IEEE Transactions on Multimedia*, vol. 6, pp. 414–422, June 2004.

[10] Hyeonuk Jeong, Jinsang Kim, and Won Kyung Cho, "Low-Power Multiplierless DCT Architecture Using Image Correlation," *IEEE Trans. Consumer Electron.*vol. 50, pp. 262–267, Feb. 2004.

[11] A. Shams, W. Pan, A. Chidanandan, and M. A. Bayoumi, "A Low-Power High Performance Distributed DCT Architecture," in *IEEE Computer Society Annual Symposium on VLSI*, Apr. 2002, pp. 21–27.

[12] L. Fanucci and S. Saponara, "Data Driven VLSI Computation For Low-Power DCT-Based Video Coding," in *International Conf. on Electronics, Circuits and Systems*, Sept. 2002, pp. 541–544.

[13] Wen Hsiung Chen, C. Smith, and S. Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform," *IEEE Trans. Commun.*, vol. 25, pp. 1004– 1009, Sept. 1977.

[14] Zhongde Wang, "Fast Algorithms for the Discrete W Transform and for the Discrete Fourier Transform,"*IEEE Trans. Acoust., Speech, Signal Processing*, vol. 32, pp. 803–816, Aug. 1984.